NBER WORKING PAPER SERIES

TEACHING ECONOMICS TO THE MACHINES

Hui Chen
Yuhan Cheng
Yanchu Liu
Ke Tang

Working Paper 34713
http://www.nber.org/papers/w34713

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
January 2026

We thank Lars Hansen, Kewei Hou, Leonid Kogan, Sai Ma (discussant), Stefan Nagel, Andrew Patton (discussant), Monika Piazzesi, Dacheng Xiu, Guofu Zhou (discussant), and participants at the NBER Conference on Big Data, Artificial Intelligence, and Financial Economics, ASSA, Econometric Society Summer School in Lausanne, CICF, AMES, SIF, MIT Sloan, Chicago Booth, Stanford GSB, UC Irvine, Tsinghua University, HKUST, Central University of Finance and Economics, Renmin University of China, and University of Melbourne for helpful comments. The views expressed herein are those of the authors and do not necessarily reflect the views of the National Bureau of Economic Research.

Teaching Economics to the Machines
Hui Chen, Yuhan Cheng, Yanchu Liu, and Ke Tang
NBER Working Paper No. 34713
January 2026
JEL No. C45, C52, G13

ABSTRACT

Structural economic models, while parsimonious and interpretable, often exhibit poor data fit and limited forecasting performance. Machine learning models, by contrast, offer substantial flexibility but are prone to overfitting and weak out-of-distribution generalization. We propose a theory-guided transfer learning framework that integrates structural restrictions from economic theory into machine learning models. The approach pre-trains a neural network on synthetic data generated by a structural model and then fine-tunes it using empirical data, allowing potentially misspecified economic restrictions to inform and regularize learning on empirical data. Applied to option pricing, our model substantially outperforms both structural and purely data-driven benchmarks, with especially large gains in small samples, under unstable market conditions, and when model misspecification is limited. Beyond performance, the framework provides diagnostics for improving structural models and introduces a new model-comparison metric based on data-model complementarity.

Hui Chen
Massachusetts Institute of Technology
MIT Sloan School of Management
and NBER
huichen@mit.edu

Yuhan Cheng
Shandong University
1004956018@qq.com

Yanchu Liu
Sun Yat-sen University
liuych26@mail.sysu.edu.cn

Ke Tang
Tsinghua University
ketang@tsinghua.edu.cn

1 Introduction

"All models are wrong, but some are useful." While acknowledging the imperfections of scientific modeling, George Box's seminal quote highlights their values in guiding us to learn from data and informing decision-making processes. However, the combination of big data and advances in machine learning techniques presents data-driven approaches as a potential new way of understanding the world without relying on traditional models and theories. One may even wonder, "Is theory dead?"[1]

Debates about the role of theory have long existed in economics. On the one hand, structural models in economics are relatively parsimonious and interpretable but tend to offer unsatisfactory fitting for empirical data. On the other hand, reduced-form models can offer significantly more flexibility and superior forecasting performances, which increasingly make them the preferred approach in prediction-oriented tasks. However, a key limitation of the reduced-form approach is that it often suffers from overfitting and lack of generalizability, especially when the sample size of training data is limited or when there is instability in the data-generating process (DGP).

In this paper, we propose a **theory-guided transfer learning** framework to integrate the economic insights from structural models with the flexibility of reduced-form machine learning models. Transfer learning, broadly speaking, aims to transfer knowledge from one context (the "source domain") to another related context (the "target domain") to improve performance in the latter. In our framework, the source domain is generated by the structural model, while the empirical data reside in the target domain. Rather than imposing potentially misspecified structural restrictions as hard constraints, we use them to inform and regularize learning in the target domain. Furthermore, because the structural model restrictions often readily extend beyond the boundaries of the training data, they also help enhance the generalizability of the reduced-form model.

Implementation is straightforward. We first train a neural network on synthetic data generated by the structural model, creating a network representation of the economic re-

[1] For example, in "The End of Theory: The Data Deluge Makes the Scientific Method Obsolete," Wired Science, 2008, the author posits, *"All models are wrong, and increasingly you can succeed without them."*

strictions. The resulting network parameters from the source-domain training then serve as a theory-informed starting point for training in the target domain, where we fine-tune the network using empirical data.

The source-domain training step benefits from two features: 1) the synthetic training set can be made arbitrarily large, limited only by computing resources; and 2) the signal-to-noise ratio in synthetic data is typically high. Coupled with the expressivity of neural networks (formally established through the universal approximation theorem; see e.g., Hornik, Stinchcombe, and White, 1989; Hanin, 2019), these features make it relatively easy to train a neural network that accurately captures the structural restrictions implied by economic theory (Chen, Didisheim, and Scheidegger, 2025).

The fine-tuning step in the target domain allows the neural network to adjust away from potentially misspecified theory and incorporate information from empirical data. Here, the objective is to minimize the empirical loss on real data. Starting with the theory-informed network parameters inherited from the source domain instead of random initialization, fine-tuning then proceeds with a learning rate that is orders of magnitude smaller than in a standard deep learning model (i.e., updating parameters in small steps), as well as a small patience parameter to limit the number of training epochs.[2] Together, these measures serve to regularize learning on real data, preventing the network from diverging too rapidly or too far from the economic restrictions. They also reduce the risk of catastrophic forgetting – a phenomenon in which neural networks abruptly "forget" previously learned information when training on new data (see e.g., Kirkpatrick et al., 2017).

This two-step procedure distinguishes our transfer learning framework from alternative approaches for integrating theoretical information into machine learning models. One such approach is to impose theoretical constraints directly during training. For example, in physics-informed neural networks (Raissi, Perdikaris, and Karniadakis, 2019), governing equations derived from physical laws are imposed as constraints on the neural network, effectively regularizing the function space the network can represent. However, since economic models are generally more prone to misspecification than their physical counterparts, imposing those

[2]The patience parameter specifies the number of epochs with no improvement on the validation set before stopping the training.

misspecified restrictions as constraints can introduce more biases in the resulting model. Achieving a desirable bias-variance tradeoff in this setting requires careful tuning of the penalty weight on the constraints, which can be challenging in practice. In contrast, while our transfer learning framework also regularizes learning on empirical data using structural restrictions, the strength of this regularization is determined implicitly by the fine-tuning process. It depends endogenously on the gap between the structural model and the true data-generating process, as well as on the sample size of the empirical data.

One can also inject theoretical information into a machine learning model using Bayesian methods. Heuristically, we can view the source-domain training as deriving a theory-informed prior for the network parameters, while the fine-tuning step effectively updates this prior using empirical data. This perspective is reminiscent of Bayesian vector autoregressions (BVAR), which imposes informative priors to help with estimating the VAR parameters.[3] However, there are three key differences between the two approaches.

First, the transfer learning framework offers vastly greater capacity to capture nonlinear relationships compared to BVARs. Second, it also enjoys a computational advantage, as performing full Bayesian updating in a high-dimensional parameter space would be prohibitively costly. Third, the two frameworks differ in how they balance the influence of the theory-implied "prior" against empirical evidence. In Bayesian settings (see e.g., the DSGE-VAR model of Del Negro and Schorfheide, 2004), this balance is governed by the ratio of the sample size of synthetic data to empirical data, a hyperparameter that needs to be tuned in the data. By contrast, in our framework, increasing the amount of synthetic data in the source domain only gives us a set of more precisely estimated network weights to initialize target-domain learning. How far the network parameters move away from this starting point is endogenously determined by fine-tuning, as explained above.

To summarize, our transfer learning framework uses economic restrictions to inform and regularize learning on empirical data. Its potential benefits can be understood through the lens of the bias-variance trade-off. The combination of theory-informed initialization and fine-tuning reduces the variance of the estimated network parameters, thereby enhancing

[3]For example, DeJong, Ingram, and Whiteman (1993); Ingram and Whiteman (1994); Del Negro and Schorfheide (2004) propose to derive an informative prior from a DSGE model.

model stability, particularly when empirical data are limited. Unlike conventional regularization methods, which typically shrink parameter estimates toward zero (e.g., via L_1 or L_2 regularization; see Hastie, Tibshirani, and Friedman, 2009), our approach shrinks them toward values implied by economic theory. At the same time, misspecification of the structural model can introduce bias into the transfer learning model. The fine-tuning step mitigates this bias by endogenously determining the relative influence of the structural model and the empirical data. A well-specified structural model provides a high-quality starting point, characterized by minimal bias, which can significantly accelerate network training and reduce the risks of becoming trapped in local minima or saddle points. Conversely, severe model misspecification may impair performance – a phenomenon known as negative transfer – when the bias introduced by the structural model outweighs the gains from variance reduction.

Besides enhancing the forecasting power, the transfer learning model can inform us about the limitations of a structural model in a few ways. First, it can help identify important features that the structural model misses. The transfer learning framework allows one to add features in the target domain beyond the state variables in a structural model, which are often kept to a small number to preserve tractability and transparency.[4] Feature importance analysis can then help identify which of these features that are left out of the structural model are important in the target domain, which would point to promising directions to improve the structural model.

Second, the transfer learning framework enables a new form of model comparison. Traditionally, we often assess a structural model's performance in isolation, based on metrics such as data fit or predictive accuracy. The transfer learning framework enables the comparison of structural models based on data-model complementarity: A structural model that performs worse on conventional metrics that focus on standalone fit may nevertheless prove more effective in guiding learning from the empirical data, if the restrictions it implies are generally aligned with the true DGP but difficult to learn directly from the training sample.

As an example application, we apply the transfer learning framework to option pricing.

[4]For example, in the Black-Scholes option pricing model, the relevant features include the stock price, strike price, time-to-maturity, risk-free rate, dividend yield, and volatility. However, other features that could be empirically relevant include past returns on the option and the underlying asset, put-call ratio, trading volume, bid-ask spreads, etc.

We use the Black-Scholes model (Black and Scholes, 1973) to generate synthetic data in the source domain. Despite ample evidence of the empirical limitations of this model, we choose it for its simplicity as well as to illustrate the point that clearly misspecified structural models can still be helpful in the transfer learning framework.

We train a feedforward neural network with 16 hidden layers and 22 neurons per layer (with a total of over 7,800 trainable parameters). In the source domain, we employ multi-target learning to capture a set of structural restrictions, with a loss function that accounts for dollar pricing errors, option delta, and vega. In the target domain, the loss function is based solely on dollar pricing errors. Source-domain training only needs to be performed once. Target-domain training and evaluation are conducted using a rolling-window approach. Each iteration uses a training and validation sample comprising three months of data, with 20% of observations randomly selected for validation, followed by a test sample from the subsequent three months. We then compare the performance of the transfer learning model (TL) against two benchmarks: (i) a deep learning model (DL) with identical architecture but without information from the structural model, and (ii) the Heston model (Heston, 1993), which extends the Black-Scholes model by incorporating stochastic volatility.

Out of sample, the TL model significantly outperforms both the DL and Heston model in pricing accuracy from 2001Q1 to 2023Q1. The average of the quarterly median absolute errors in Black-Scholes implied volatility (BSIV-MAE) for the TL model in the entire sample is 32.4% of that for DL and 9.3% of that for the Heston model.

In the cross section, the performance advantage of TL over DL is particularly pronounced for median and long time-to-maturity, out-of-the-money, and more liquid options (those with narrower bid-ask spreads or higher trading volume). In the time series, the advantage widens when market volatility is elevated (measured by the average VIX over the preceding 60 days), when volatility spikes, and when new observations appear uncommon relative to the training data (measured by the Mahalanobis distance between their input features and the training sample). These results are consistent with the interpretation that structural model restrictions are more helpful when they are less misspecified (for example, for longer-dated or more liquid options), when generalization beyond the training data is required, or when we need to adapt to the instability in the underlying DGP.

The TL model also demonstrates greater stability compared to the DL model, in two key dimensions. First, the outliers of TL pricing errors are less extreme. For example, the 90th percentile of out-of-sample absolute BSIV errors for the TL model is 34% of that for the DL model over the full sample. Second, neural networks inherently contain randomness from the training process – such as stochastic gradient descent, parameter initialization, and sampling uncertainty. Along this dimension, the TL model also shows markedly lower dispersion in outcomes, primarily because it uses theory-implied initialization and a small learning rate in the target domain. The stability of the TL model is particularly helpful for small-sample learning. For example, when we reduce the training sample to 10% of its original size in each three-month period, the performance of the DL model deteriorates sharply, whereas that of the TL model remains relatively stable.

Since the TL model is trained on both synthetic and empirical data, one may wonder whether the DL model could achieve comparable performance if trained on more data. We expand the DL training sample in two ways: (i) by pooling the synthetic and empirical data, and (ii) by switching from the 3-month rolling window to an expanding window starting from 2000Q4. The results show that naive data pooling does not improve DL performance. Although adding synthetic training data can reduce variance for the DL model, it also introduces bias when the structural model is misspecified. The relative sample size of synthetic vs. empirical data is therefore critical for the bias-variance trade-off: too much or too little synthetic data can both lead to suboptimal performance. The expanding-window approach does improve DL accuracy, but the performance gap relative to TL remains large.[5] This latter exercise underscores a key challenge with time-series forecasting: when the DGP can change over time, observations from the distant past become less informative. Thanks to its ability to achieve stable performance on small datasets, the TL model is better equipped to adapt to structural breaks and distributional shifts.

We also examine two alternative approaches for incorporating information from the structural model in this option pricing experiment. The first mimics physics-informed neural networks by imposing the structural restrictions as constraints during DL training.

[5]Under the expanding-window setting, the out-of-sample BSIV-MAE for the DL model falls to 62% of that for the original DL under rolling-window training.

Specifically, we add the average pricing errors relative to the Black-Scholes model as a penalty term in the loss function, with the penalty weight tuned using data. The second approach fits a linear model to the pricing errors of the Black-Scholes model to boost its predictive performance. Neither approach matches the performance of the TL model. Unlike the constrained approach, the TL framework uses the (potentially misspecified) structural model for informed regularization rather than imposing it as a hard constraint. The boosting method also builds on the structural model, but its training on the structural model errors remains prone to overfitting. In fact, the overfitting issue might be even more severe due to the lower signal-to-noise ratio in the pricing errors compared to the prices themselves.

Finally, it is worth noting that our transfer learning framework is quite general. It can be applied to a wide range of forecasting problems whenever a suitable structural model is available to generate synthetic data. Moreover, the framework is not limited to neural networks – the two-step procedure can be applied to a variety of parametric models, such as generalized linear models, Bayesian models, etc. The benefits of incorporating structural model restrictions are likely to be most pronounced when the underlying machine learning model has high flexibility relative to the size and representativeness of the training sample.

Related literature There is a fast-growing literature that applies machine learning methods to economics and finance. Among the early contributions are Hutchinson, Lo, and Poggio (1994), Chen and White (1999), and Chen and Ludvigson (2009). Recent works have shown the rich benefits of machine learning techniques in both linear (see e.g., Kelly and Pruitt, 2013; Rapach and Zhou, 2013; Chinco, Clark-Joseph, and Ye, 2019; Kozak, Nagel, and Santosh, 2023) and nonlinear settings (see Gu, Kelly, and Xiu, 2020; Freyberger, Neuhierl, and Weber, 2020; Bianchi, Büchner, and Tamoni, 2021; Cong et al., 2022; Bali et al., 2023; Chen, Pelger, and Zhu, 2024; Campello, Cong, and Zhou, 2024, among others).

It is intuitive that one should try to take advantage of domain knowledge when applying these powerful methods. In practice, this often involves using economically motivated features and feature engineering in forecasting models. Several studies have gone further by incorporating theoretically motivated restrictions. For example, Garcia and Gençay (2000) show that the performance of neural networks can be improved by exploiting the

homogeneity implied by the option pricing formula. The no-arbitrage condition has been used by Feng et al. (2023), Chen, Pelger, and Zhu (2024), Bryzgalova, Pelger, and Zhu (2023), and Bryzgalova et al. (2024) to help estimate factor betas, identify weak factors, and estimate the conditional stochastic discount factor. Our contribution is to propose a general transfer learning framework that uses potentially misspecified structural model restrictions to guide the learning on empirical data.[6]

There are several alternative approaches for incorporating theoretical restrictions into econometric models. First, a strand of Bayesian Vector Autoregression models integrate prior information based on economic theories, with the goal of reducing overfitting in high-dimensional models through coefficient shrinkage. These prior can be statistically motivated (as in the case of the Minnesota prior, see Doan, Litterman, and Sims, 1984; Litterman, 1986) or derived from a structural model (see DeJong, Ingram, and Whiteman, 1993; Ingram and Whiteman, 1994; Del Negro and Schorfheide, 2004). Second, in physics-informed neural networks (Raissi, Perdikaris, and Karniadakis, 2019), structural model restrictions are treated as hard constraints and incorporated into the loss function. We face bias-variance tradeoffs when these structural restrictions are likely misspecified, which could make transfer learning the more suitable approach in such situations. Third, Almeida et al. (2023) propose to boost parametric option pricing models by fitting a neural network on the model-implied pricing errors. We investigate a simplified version of this boosting approach, replacing the neural network with a linear model.

2 Methodology

Transfer learning is based on the idea that knowledge acquired from solving one problem in a source domain can be transferred to improve learning in a related but distinct target domain (e.g., Caruana, 1997; Weiss, Khoshgoftaar, and Wang, 2016). The knowledge obtained in the source domain not only makes it easier to train a model in the target domain, especially when data are limited, but could also improve the model's performance. In our setting, extracting

[6]Transfer learning is widely used technique in machine learning. Important applications include computer vision and large language models. For a comprehensive survey on this topic, see Zhuang et al. (2020).

information from an economic model constitutes the source task, and this information is then used to aid the target task of learning directly from empirical data.

2.1 A Theory-guided Transfer Learning Framework

Denote by $\mathcal{X}$ and $\mathcal{Y}$ the input and target space, respectively, with unknown probability distribution $\mathbb{P}_{(\mathcal{X},\mathcal{Y})}$ on some σ-algebra of $\mathcal{X} \times \mathcal{Y}$. Our goal is to find $f(x)$ that minimizes the expected loss $\mathbb{E}^{\mathbb{P}}[\mathcal{L}(f(x), y)]$ for some loss function $\mathcal{L}$. In a standard supervised learning setting, we search for f from within a given hypothesis class $\mathcal{H}$ (e.g., the set of neural networks) that minimizes the empirical loss

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f(x_i), y_i)$$

over a given training set $S = ((x_i, y_i))_{i=1}^n$. The quality of the learned function f depends on several factors: i) the flexibility of the class $\mathcal{H}$, ii) the randomness of the training sample, and iii) the training procedure (e.g., random initialization of search and stochastic gradient descent).

While our framework is more general, to fix ideas, we will focus on the case where the set $\mathcal{H}$ consists of neural networks in the remainder of this paper. A neural network with L layers is denoted as $F(L; \sigma_1, \sigma_2, \cdots, \sigma_L; W_1, W_2, \cdots, W_L)$, where σ_i and W_i are the activation function (a non-linear function that is applied element-wise) and weights for the i-th layer, respectively. The training of the neural network results in weights $W^* = [W_1^*, W_2^*, \cdots, W_L^*]$.

Next, consider an economic structural model that imposes certain restrictions between x and y. The corresponding joint distribution of (x, y), which may not be fully specified, is $\mathbb{Q}_{(\mathcal{X},\mathcal{Y})}$. We assume that the structural model either specifies the conditional distribution of y given x, or, at a minimum, the conditional expectation of y given x,

$$\mathbb{E}^{\mathbb{Q}}[y \mid x] = g(x), \tag{1}$$

where the expectation is taken under the model-implied probability measure $\mathbb{Q}$. The structural model may be misspecified in the sense that $\mathbb{Q}$ is not consistent with the true data-generating

process $\mathbb{P}$.

We propose a transfer learning framework that uses the structural model to guide the training of the machine learning model. It involves two steps. First, in the source domain, we generate a synthetic dataset $\widetilde{S} = ((\tilde{x}_i, \tilde{y}_i))_{i=1}^{\tilde{n}}$ from the structural model and train a neural network by minimizing the empirical loss over $\widetilde{S}$ with respect to a loss function $\widetilde{\mathcal{L}}$. Let the resulting network be $\widetilde{f}$, with weights $\widetilde{W}^*$. In the second step, we fine-tune $\widetilde{f}$ using the empirical dataset S. Specifically, we initialize the network weights with $\widetilde{W}^*$ obtained from the source domain and minimize a loss function $\widehat{\mathcal{L}}$ that is potentially different from its source-domain counterpart, $\widetilde{\mathcal{L}}$. The resulting neural network is denoted by $\widehat{f}$, with weights $\widehat{W}^*$. Notice that, for clarity, we use the notation $\widetilde{\cdot}$ to denote quantities associated with the source domain and $\widehat{\cdot}$ for those associated with the target domain.

Next, we discuss these two steps in more detail.

Source Domain Our goal in the source domain is to obtain a neural network representation of the structural model restrictions. As shown by Chen, Didisheim, and Scheidegger (2025), neural networks are well-suited for this task thanks to their expressivity. In particular, the universal approximation theorem establishes that a sufficiently wide or deep neural network can approximate any smooth function to arbitrary accuracy (see Hornik, Stinchcombe, and White, 1989; Hanin, 2019). This step further benefits from two practical advantages: the synthetic dataset can be made arbitrarily large (i.e., $\tilde{n} \gg n$, which is limited only by computational resources), and the signal-to-noise ratio of synthetic data is typically high. Together, these properties substantially reduce the risk of overfitting in the source domain.

Economic models are typically designed to be parsimonious, with a relatively small number of state variables. In contrast, a main attraction of machine learning models is that they help us look for information in many potential features empirically, often extending far beyond those considered relevant in an economic model. At the same time, a structural model may involve latent states or parameters that are not directly observable in the empirical data, i.e., not included in $\mathcal{X}$. Thus, as a first step, feature augmentation may be needed to align the input vectors in the source domain and target domains so that they match in dimension.

Denote by x^S the vector of relevant inputs (including states and parameters) for the

structural model. In the source domain, we augment x^S with the additional features used in the target domain. Similarly, in the target domain, we augment the input vector x with the additional inputs required by the structural model. Formally, we define the source-domain projection function

$$\Phi_S(x_i^S) \equiv \langle P_C x_i^S, P_S x_i^S, e_i^S \rangle, \tag{2}$$

where P_C is a partial identity matrix (also known as selection matrix) with values of 1 corresponding to the common features between x and x^S, P_S is a selection matrix with values of 1 for the features in x^S but not in x, and e_i^S is a vector with the same dimension as the subset of features in x but not in x^S. Similarly, we define the target-domain projection function

$$\Phi_T(x_i) \equiv \langle Q_C x_i, e_i^T, Q_T x_i \rangle, \tag{3}$$

with Q_C and Q_T defined analogously to P_C and P_S, and e_i^T a vector with the same dimension as the subset of features in x^S but not in x.

The vector e_i^S, corresponding to features considered irrelevant by the structural model, can be drawn from a certain distribution or simply set to zero. The vector e_i^T includes the hidden states and parameters that are required by the structural model but are not directly observable in the empirical data. We can either estimate them in the empirical data using the structural restrictions, or integrate them out – for example, by integrating the conditional expectation of the target variable y in Eq. (1) over some hierarchical prior for the parameters and the model-implied distribution of the hidden states. It is worth noting that the neural network in the source domain, if trained accurately, can be used to accelerate the estimation of parameters and hidden states significantly (see Chen, Didisheim, and Scheidegger, 2025). For notational simplicity, we continue to refer to the augmented feature vector as x.

Next, to generate the synthetic dataset $\widetilde{S} = ((\tilde{x}_i, \tilde{y}_i))_{i=1}^{\tilde{n}}$, we first draw $\tilde{x}_i$ and then either draw $\tilde{y}_i$ from the conditional distribution of y given x implied by the structural model, or directly set it to the conditional expectation of y given x according to Eq. (1). To sample $\tilde{x}_i$, we begin by specifying the ranges of the relevant state variables and parameters for the

structural model and then draw their values from the multivariate uniform distribution over the specified domain.[7] Finally, following Eq. (2), we augment the model features with the "irrelevant" features e_i^S to complete the input vector $\tilde{x}_i$.

How should we specify a loss function for source-domain training? Naturally, we would like to minimize some measure of the distance between the network prediction $\widetilde{f}(\tilde{x}_i)$ and the target variable $\tilde{y}_i$, for example, by using the mean squared error (MSE) over the synthetic training set. Different from a standard supervised learning setting, where the DGP is unknown, the structural model may provide additional restrictions that can be informative for the neural network. Consider the example of training a physics-informed neural network for a data-driven solution to the heat equation. As Raissi, Perdikaris, and Karniadakis (2019) show, in addition to matching the initial and boundary data, the residuals of the heat equation can be part of the loss function, which is referred to as physics-informed loss. Similarly, we can add economics-informed loss terms to the source domain loss function.

Specifically, in the source domain, we train the neural network over the synthetic dataset $\widetilde{S} = ((\tilde{x}_i, \tilde{y}_i))_{i=1}^{\tilde{n}}$ by

$$\widetilde{W}^* = \arg\min_{\widetilde{W}} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \left(\lambda_0 \widetilde{\mathcal{L}}_0(F(\tilde{x}_i; \widetilde{W}), \tilde{y}_i) + \sum_{j=1}^{K} \lambda_j \widetilde{\mathcal{L}}_j(F(\tilde{x}_i; \widetilde{W})) \right). \tag{4}$$

The first term of the loss function, $\widetilde{\mathcal{L}}_0$, measures the discrepancy between the network prediction and the target variable in the synthetic dataset. For example, it could be the absolute error for the network's predictions,

$$\widetilde{\mathcal{L}}_0(F(\tilde{x}_i; \widetilde{W}), \tilde{y}_i) = \left| F(\tilde{x}_i; \widetilde{W}) - \tilde{y}_i) \right|. \tag{5}$$

The economics-informed loss terms are given by $\widetilde{\mathcal{L}}_j$ $(j = 1, \cdots, K)$. They are derived from the additional structural restrictions in the economic model. For example, we can use the

[7] The uniform distribution ensures that the synthetic data reflect the structural model's restrictions across the entire feature space. Alternative sampling strategies may also be used – for example, oversampling regions where the structural relationships are highly nonlinear.

model-implied gradient of $g(x)$ from Eq. (1) as part of the economics-informed losses,

$$\widetilde{\mathcal{L}}_j(F(\tilde{x}_i;\widetilde{W})) = \left|\frac{\partial F}{\partial x_j}(\tilde{x}_i;\widetilde{W}) - \frac{\partial g}{\partial x_j}(\tilde{x}_i)\right|. \tag{6}$$

The weights assigned to each loss term, $(\lambda_0, \cdots, \lambda_K)$, are hyperparameters that can be tuned using cross-validation.

Training begins with random initialization of the weight matrix $\widetilde{W}$. The neural network is then trained by minimizing the loss function via stochastic gradient descent. Because the synthetic dataset can be made arbitrarily large and typically exhibits a high signal-to-noise ratio, the risk of overfitting is low. Consequently, we can employ a relatively high learning rate and a large number of epochs to ensure convergence to the optimal weights $\widetilde{W}^*$.

Target Domain In the target domain, we fine-tune the network obtained from the source domain over the empirical dataset S. Specifically, we minimize the empirical loss with respect to the loss function $\widehat{\mathcal{L}}$ over the empirical dataset $S = ((x_i, y_i))_{i=1}^n$,

$$\widehat{W}^* = \arg\min_{\widehat{W}} \frac{1}{n}\sum_{i=1}^{n} \widehat{\mathcal{L}}(F(x_i;\widehat{W}), y_i), \tag{7}$$

with initial weights set to the optimal weights $\widetilde{W}^*$ inherited from the source domain Eq. (4). An example of the loss function $\widehat{\mathcal{L}}$ could be the mean absolute prediction error.

The fine-tuning process is characterized by a substantially smaller learning rate than in the source domain and a low patience parameter for early stopping. Together, these settings make it more difficult for the final weights $\widehat{W}^*$ to deviate significantly from $\widetilde{W}^*$. This design has two benefits. First, it reduces the risk of catastrophic forgetting for the neural network – the loss of structural model information acquired in the source domain. Second, it effectively regularizes the network training in the target domain, not by shrinking the weights towards zero, but towards the theory-informed initial weights $\widetilde{W}^*$.

Note that an alternative way to reduce the risk of catastrophic forgetting is explicit weight regularization, which adds a regularization term to the loss function in (7) that penalizes deviations of the weights from their source-domain values (see e.g., Li, Grandvalet, and

Davoine, 2018),

$$\widehat{W}^* = \arg\min_{\widehat{W}} \frac{1}{n} \sum_{i=1}^{n} \hat{\mathcal{L}}(F(x_i; \widehat{W}), y_i) + \lambda \mathcal{R}(\widehat{W}; \widetilde{W}^*), \tag{8}$$

where $\mathcal{R}$ is the regularization term, with parameter $\lambda \geqslant 0$ controlling the regularization strength. We could, for example, impose an L_2 penalty centered at the theory-informed weights $\widetilde{W}^*$, $\mathcal{R}(W) = \|W - \widetilde{W}^*\|_2^2$. Considering the permutation symmetry of network weights,[8] we can also penalize the differences in network outputs relative to the structural model rather than the network weights. In this setting, the regularization parameter λ needs to be tuned separately, for example, via cross-validation. By contrast, with our fine-tuning approach the regularization effect arises implicitly from the use of a small learning rate and early stopping. Its effective strength depends endogenously on the gap between the structural model and the true DGP, as well as the sample size of empirical data. We compare these two approaches in our empirical exercise in Section 5.1.

In the transfer learning literature, various ad hoc strategies have been proposed to preserve information from the source domain through architectural design. One common approach is partial fine-tuning, which freezes the first $K < L$ layers of the source-domain model and updates only the remaining layers. Another strategy is to add new layers – often referred to as a "network head" – on top of the pre-trained model. These additional layers are initialized randomly and trained jointly with the existing network, allowing the model to adapt to features that are specific to the target domain while retaining the representations learned from the source domain. In this paper, we focus on the full fine-tuning method for its simplicity, and leave the exploration of these alternative strategies to future research.

2.2 Bias-variance Trade-off for Theory-guided Transfer Learning

Having explained the general framework, we now use a simple example to illustrate the bias-variance trade-off in the context of theory-guided transfer learning.[9] Suppose the true

[8] Since the parameterization of neurons in a network is not unique (see e.g., Entezari et al., 2021), it may not be as meaningful to treat $\widetilde{W}^*$ as a single anchoring point.

[9] We thank Sai Ma for suggesting this example.

data-generating process ($\mathbb{P}_{(\mathcal{X},\mathcal{Y})}$) is given by

$$y_i = x_{1i} + x_{2i} + x_{1i}x_{2i} + \epsilon_i, \tag{9}$$

where x_{1i} and x_{2i} are independent standard normal random variables; ϵ_i is IID normal, $\epsilon_i \sim N(0, \sigma^2)$. The parameter σ controls the signal-to-noise ratio of the data. The theoretical lower bound for the out-of-sample MSE of any model based on x_{1i} and x_{2i} is σ^2.

Next, consider a potentially misspecified "theoretical model" ($\mathbb{Q}_{(\mathcal{X},\mathcal{Y})}$) of the data,

$$y_i = x_{1i} + x_{2i} + (1-m)x_{1i}x_{2i}, \tag{10}$$

where the coefficient $m \in [0, 1]$ controls the degree of misspecification, or DoM, of the theoretical model (the model is correctly specified when $m = 0$; it completely misses the nonlinear term when $m = 1$).

To build the TL model, we first generate synthetic data using the theoretical model (10) and train a neural network in the source domain,[10] and then fine-tune it using an empirical dataset, which is simulated from (9). The sample size of empirical data is n. For comparison, we also train a DL model with identical architecture directly on the same sample of empirical data. The learning rate for TL is set to 10^{-5} in the fine-tuning step, while for DL it is set to 10^{-3}, and both models utilize an early stopping strategy.[11] We then examine the out-of-sample MSE (computed over 50,000 new test samples drawn from Eq. (9)) for the two models as a function of the degree of misspecification m, the signal-to-noise ratio σ, and the empirical-data sample size n.

The results are shown in the first row of Figure 1. Panels A and B, with $\sigma = 1$, correspond to cases where the data have relatively high signal-to-noise ratio. In Panels C and D, the data are more noisy, with $\sigma = 5$. In Panels A and C, we set the sample size of empirical data to $n = 100$ to illustrate the setting where data are relatively scarce. In Panels B and D, the

[10] In this experiment, we use a feedforward neural network with two hidden layers of 64 neurons each and a ReLU activation function. The sample size of synthetic data is $\tilde{n} = 5,000$.

[11] The empirical dataset is split into training and validation sets with a 9:1 ratio. The training procedure is terminated if the loss function on the validation set fails to improve for 5 consecutive iterations or once the number of epochs reaches 200.

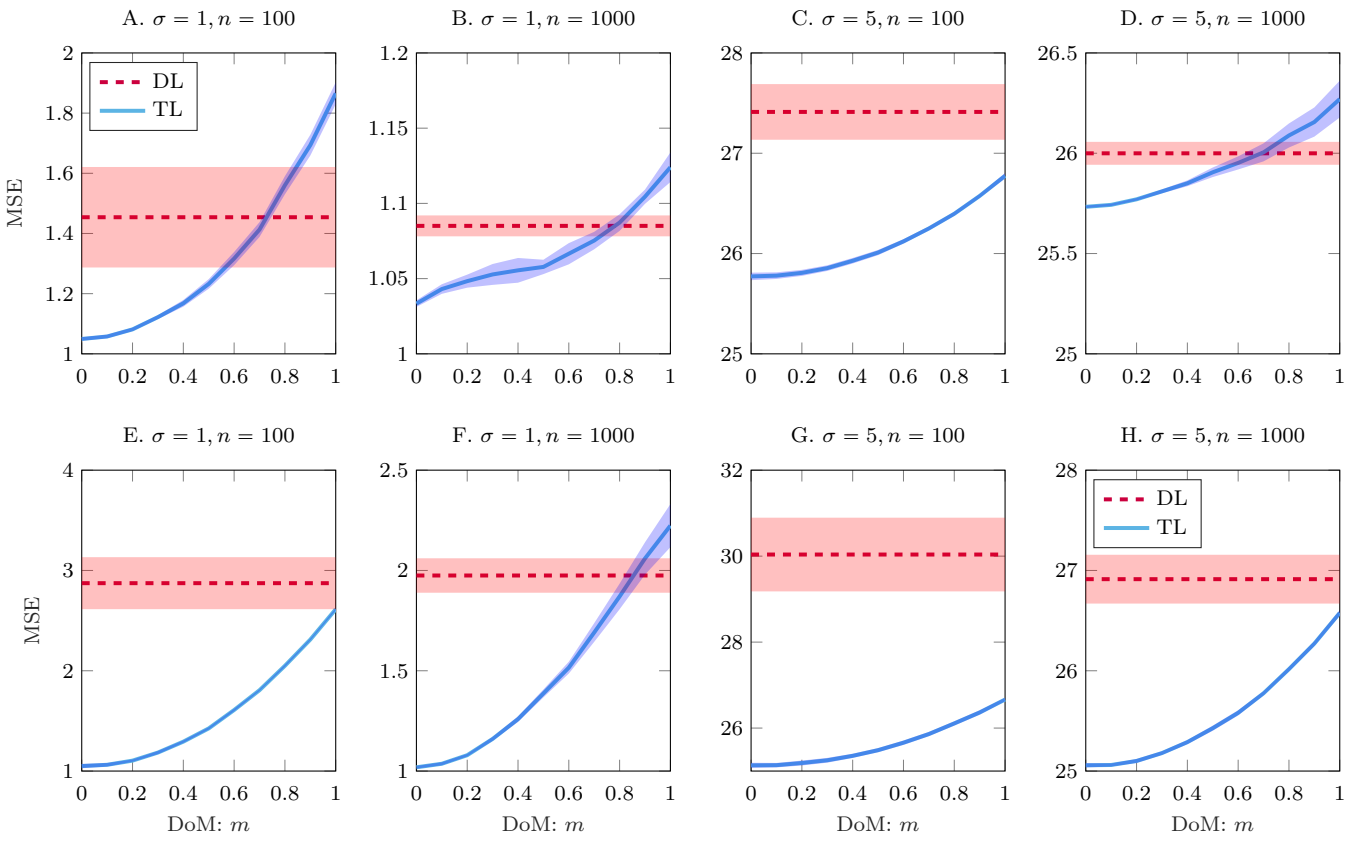

Figure 1: **Illustration of the bias-variance trade-off for TL.** This figure reports the out-of-sample MSE for TL and DL from the simple example. The blue solid line represents the MSE for TL, and the red dashed line represents the MSE for DL. The shaded regions represent 95% confidence intervals based on 40 simulations.

empirical sample size is raised to $n = 1,000$.

Across all four panels, the performance of the TL model deteriorates as the degree of misspecification m in the theoretical model increases. This result is intuitive: model misspecification introduces a bias into the TL model. When the degree of misspecification becomes sufficiently large, the bias can outweigh the benefit of theory-informed regularization (via the theory-informed initial weights and fine-tuning), resulting in negative transfer and causing the TL model to underperform the DL model. This pattern is evident in the figures, where the MSE for TL (blue solid line) sometimes exceeds that for DL (red dashed line) as m approaches one.

Notice also the wide ranges of m across the four panels (A through D) within which the MSE for TL remains below that for DL. This suggests that the TL model can outperform

the DL model even when guided by a misspecified theoretical model, particularly when the training data are noisy and limited in size. For example, in Panel C (with a large σ and small n), the MSE for TL stays below that for DL across all values of m. In such settings, standard DL models are especially prone to overfitting, making the benefits of theory-informed regularization more pronounced.

Holding σ fixed, as the training sample size n increases, the performance gap between DL and TL shrinks under different values of m. The same is true if we hold n fixed and decrease σ. These results are consistent with the intuition that information from the theoretical model becomes less valuable in data-rich environments.

A natural question is whether conventional weight regularization techniques, such as L_1 or L_2 regularization, can enable a DL model to achieve performance gains comparable to those achieved through theory-guided transfer learning. The answer is no. For example, when an L_2 penalty is applied to the network weights of the DL model, with the penalty parameter selected via cross-validation, the out-of-sample MSE improves only marginally in three of the four cases considered above, with no meaningful improvement in the fourth case. Even the largest reduction in the MSE is still below 2.5%, much smaller than what TL achieves for a wide range of m values. These results suggest that the theory-informed regularization, even when based on a misspecified theoretical model, can be substantially more effective than conventional shrinkage toward zero.

Out-of-distribution generalization. We also use the same example to examine the differences between the TL and DL models in out-of-distribution (OOD) generalization – that is, the ability of a model to generate accurate predictions for data drawn from distributions not observed during training. While deep neural networks are highly effective interpolators, they are known to perform poorly when extrapolating beyond the support of their training data. In economic and financial settings, however, such extrapolation is often unavoidable: test data may fall outside the range of the training data due to large shocks or limited historical samples.

To assess whether theory-guided transfer learning can enhance generalization in such situations, we restrict the training observations to those with (x_{1i}, x_{2i}) lying within the unit

circle (i.e., $x_{1i}^2 + x_{2i}^2 \leqslant 1$), while drawing the test observations from outside the unit circle. The out-of-sample accuracy therefore directly measures a model's ability to extrapolate relationships learned from within the unit circle to regions beyond it. All other aspects of model training and evaluation remain unchanged.

The results, shown in the second row of Figure 1 (Panels E through H), confirm that the TL model indeed generalizes better than the DL model. While the performance of both models declines relative to the in-distribution case, the deterioration is much more pronounced for DL across all four panels. Except for Panel F, where negative transfer still appears in a narrow region where m is close to 1, the TL model outperforms the DL model across all m values and by a considerably larger margin than in their counterparts in the row above.

Beyond illustrating how theory-guided transfer learning can be understood through the lens of the bias-variance trade-off, the results from this simple example also underscore that the value of a theoretical model within the transfer learning framework cannot be judged solely by its goodness of fit. In environments with limited or noisy data, even a substantially misspecified theoretical model can still help us learn more effectively from the data.

3 Empirical Results

In this section, we use the problem of pricing S&P 500 index options to examine the properties of the theory-guided transfer learning framework and compare it with standard machine learning models. We begin by describing how the framework is implemented in the option-pricing context, then discuss the data and evaluate the pricing performance of the transfer learning model. Finally, we illustrate how an attribution analysis can identify the key variables driving model performance and point to potential directions for improving the structural model.

3.1 Applying TL to Option Pricing

To build a transfer learning model for option pricing, we first need to specify a structural model for the source domain. We select the Black-Scholes model for two main reasons. First, as the seminal option pricing model, it is arguably the least subject to "look-ahead" biases

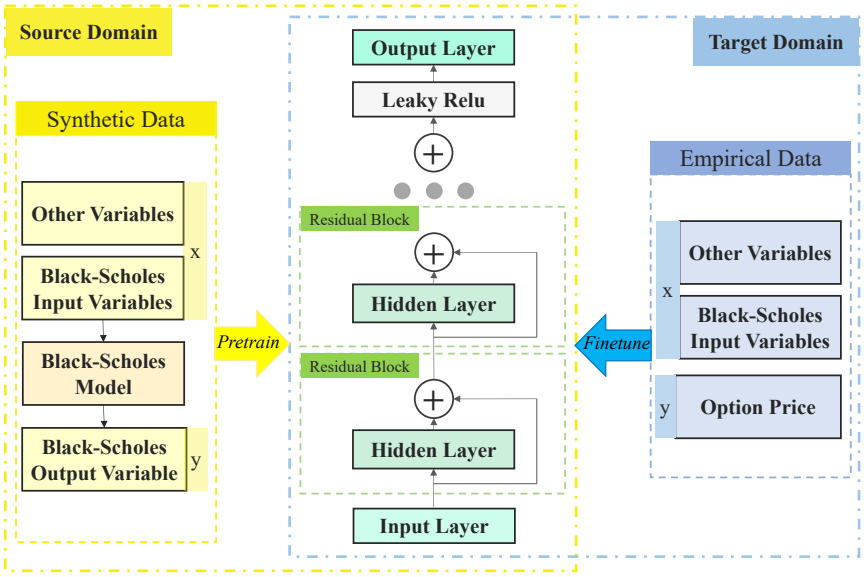

Figure 2: **The Transfer Learning Framework.** This figure illustrates the application of our adopted transfer learning framework in the context of option pricing.

(Black and Scholes, 1973, was published before the introduction of S&P 500 index options in 1983). Since subsequent models build on the Black-Scholes model by addressing its limitations in the data, using them as the source domain could make it difficult to isolate the benefits of transfer learning. Second, it helps demonstrate that even a relatively simple and misspecified structural model can provide useful guidance within the transfer learning framework.

The diagram in Figure 2 illustrates the transfer learning framework for option pricing. In the source domain, we use a deep residual multilayer perceptron (ResMLP) with $LeakyReLU$ activation, which has 16 hidden layers, 16 neurons in each layer. We train the model on a synthetic dataset of 800,000 observations simulated from the Black-Scholes model (see Appendix A for details). The source domain training is performed only once. In the target domain, we fine-tune the same pretrained source domain model using the empirical data.

The feature vector for the option pricing model is: $x_i = (S_i, K_i, \tau_i, vol_i, d_i, r_i, Z_i)$, where the first six variables – stock price S_i, strike price K_i, time-to-maturity τ_i, volatility vol_i (proxied by the one-day-lagged VIX index), dividend yield d_i, and risk-free rate r_i – are those required by the Black-Scholes model. The additional features, represented by Z_i, include the historical volatility of S&P 500 returns, short- and medium-term momentum of both the

S&P 500 index and the specific option (based on one-week and four-week returns), abnormal option trading volume, put-call ratio, and the S&P 500 earnings-price ratio. The complete list of features is provided in Appendix B.

For feature augmentation in the target domain, all six variables required by the Black-Scholes model are directly observable in the empirical data (including the volatility parameter, which we proxy with lagged VIX index) and thus do not need to be estimated. In the source domain, we draw the features (including those in Z_i) from independent uniform distributions, with the details specified in Table A.1 in Appendix B. Corresponding to each synthetic feature vector $\tilde{x}_i$, we compute the option price $\tilde{y}_i$, option Delta $\frac{\partial g(\tilde{x}_i)}{\partial S}$, and Vega $\frac{\partial g(\tilde{x}_i)}{\partial vol}$ according to the Black-Scholes model.

In the source domain, we train the neural network over the synthetic dataset $\widetilde{S} = ((\tilde{x}_i, \tilde{y}_i))_{i=1}^{\tilde{n}}$:

$$\widetilde{W}^* = \arg\min_{\widetilde{W}} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \left(\lambda_0 \widetilde{\mathcal{L}}_0(F(\tilde{x}_i; \widetilde{W}), \tilde{y}_i) + \lambda_1 \widetilde{\mathcal{L}}_1(F(\tilde{x}_i; \widetilde{W})) + \lambda_2 \widetilde{\mathcal{L}}_2(F(\tilde{x}_i; \widetilde{W})) \right), \tag{11}$$

where

$$\widetilde{\mathcal{L}}_0(F(\tilde{x}_i; \widetilde{W}), \tilde{y}_i) = w_i \left| F(\tilde{x}_i; \widetilde{W}) - \tilde{y}_i \right|, \tag{12a}$$

$$\widetilde{\mathcal{L}}_1(F(\tilde{x}_i; \widetilde{W})) = \left| \frac{\partial F(\tilde{x}_i)}{\partial S} - \frac{\partial g(\tilde{x}_i)}{\partial S} \right|, \tag{12b}$$

$$\widetilde{\mathcal{L}}_2(F(\tilde{x}_i; \widetilde{W})) = \left| \frac{\partial F(\tilde{x}_i)}{\partial vol} - \frac{\partial g(\tilde{x}_i)}{\partial vol} \right|, \tag{12c}$$

with randomly initialization for the network weights $\widetilde{W}$.

The first term in the loss function (11), $\widetilde{\mathcal{L}}_0$, is the weighted mean-absolute dollar pricing error. Since the dollar prices (and dollar pricing errors) can vary significantly with option moneyness, we specify weights w_i to ensure that the TL model does not overlook out-of-the-money (OTM) options. Since our empirical analysis focuses exclusively on put options, we

set the weight for contract i to be:[12]

$$w_i = \frac{1}{1 + Delta_i + \epsilon_c}. \tag{13}$$

A small positive constant ϵ_c is added to avoid exploding weights.

In addition to minimizing pricing errors relative to the Black-Scholes model, we also add two economics-informed loss terms, $\widetilde{\mathcal{L}}_1$, which matches the network's prediction about Delta with the Black-Scholes model, and $\widetilde{\mathcal{L}}_2$, which matches the network's prediction about Vega with the Black-Scholes model. These restrictions provide further information about the option pricing function in the structural model, specifically its sensitivity to changes in the stock price and volatility, to the neural network.

In the target domain, we set the loss function to be the weighted mean absolute dollar pricing errors:

$$\widehat{W}^* = \arg\min_{\widehat{W}} \sum_{i=1}^{n} w_i \left| F(x_i; \widehat{W}) - y_i \right|, \tag{14}$$

where the weights w_i are again inversely related to the Black-Scholes option delta (see (13)). The network weights initialized using $\widetilde{W}^*$ inherited from the source domain. As explained in Section 2, the fine-tuning process in the target domain uses a significantly smaller learning rate than the one used in source-domain training and a low patience parameter. For details on the network architecture and hyperparameters used in both domains, please refer to Appendix A.

3.2 Data

We use daily transaction data of the S&P 500 (European) index options from OptionMetrics, covering the period from January 2, 2000, to March 31, 2023. Due to the fact that the put-call parity holds reasonably well for SPX options in the data, we focus exclusively on pricing put options. Table 1 reports summary statistics for this dataset, which includes a total of 27,791,709 contract-day observations. We categorize options by three time-to-maturity (τ)

[12] Put options have negative delta, with delta ranging from near 0 for deep ITM puts to near -1 for deep OTM puts.

Table 1: **Descriptive statistics**

In this table, we divide individual SPX put option contracts into 18 categories based on time-to-maturity and moneyness. Within each category, it reports the number of contract-day observations (N), the sample mean of Black-Scholes implied volatility (IV-Mean), and the standard deviation of Black-Scholes implied volatility (IV-STD). Sample period: January 2, 2000, to March 31, 2023.

	N (million)			IV-Mean			IV-STD		
$\ln(K/S)$	$\tau < 10$	$10 \leqslant \tau \leqslant 60$	$\tau > 60$	$\tau < 10$	$10 \leqslant \tau \leqslant 60$	$\tau > 60$	$\tau < 10$	$10 \leqslant \tau \leqslant 60$	$\tau > 60$
< -0.5	0.256	0.920	1.710	0.700	0.794	0.734	0.374	0.319	0.378
$[-0.5, -0.25]$	0.469	1.710	1.560	0.725	0.727	0.628	0.377	0.381	0.416
$[-0.25, 0]$	2.510	6.490	3.240	0.538	0.504	0.401	0.433	0.396	0.301
$[0, 0.25]$	1.510	3.930	2.540	0.144	0.121	0.175	0.213	0.156	0.236
$[0.25, 0.5]$	0.073	0.234	0.500	0.515	0.309	0.268	0.452	0.324	0.389
> 0.5	0.005	0.029	0.108	0.547	0.356	0.333	0.510	0.403	0.432

bins and six moneyness ($\ln(K/S)$) bins, yielding 18 distinct subsamples.

The patterns for Black-Scholes implied volatility (calculated based on mid-prices) across different moneyness bins are consistent with the well-documented volatility smile: contracts that are in- or out-of-the-money exhibit higher implied volatilities than those that are at-the-money, especially for the out-of-the-money puts. Since the Black-Scholes model cannot capture the volatility smile, it is clearly misspecified and leaves room for improvement within the transfer learning framework. It is also worth noting that the sample sizes vary significantly across different moneyness and maturity bins. For example, deep out-of-the-money options (with $\ln(K/S) < -0.5$) and deep in-the-money options (with $\ln(K/S) > 0.5$) have considerably smaller sample sizes, especially when the time-to-maturity is short (with $\tau < 10$ days). This feature will affect the relative performance of the transfer learning model and standard machine learning models.

3.3 Pricing Performance

We use a rolling-window scheme to train and evaluate the TL model. By treating each quarter as a separate training episode, we can assess the model's performance and robustness under various market conditions. At the end of each quarter, we start with the same source-domain model and use the empirical data from the preceding three months for target-domain fine-tuning. The three-month dataset is split into training and validation sets using an

80:20 ratio. The validation set is used to implement early stopping, enabling the algorithm to automatically select the optimal number of training epochs. The empirical data from the subsequent three months serve as the test set to evaluate the model's out-of-sample performance.

To compare the TL model with traditional methods, we construct two benchmark models. The first is a conventional deep learning (DL) model. This model uses the same training data, network architecture, activation function, and stopping rule as the TL model. The only differences are the network initialization and the learning rate. Specifically, instead of theory-informed initialization, the DL model employs standard random initialization. Moreover, we allow the TL and DL models to use their own optimal learning rates, which are determined via a grid search on a training-validation split based on the data preceding the first training-testing period (i.e., from 2000Q1 to 2000Q3).[13] The second benchmark is the stochastic volatility model of Heston (1993), whose structural parameters are estimated in a rolling fashion by minimizing the training-set pricing error over the full three-month sample.

Intuitively, evaluating model performance using dollar-price-based errors may overweight option contracts with higher price levels (e.g., in-the-money or longer-maturity options). To ensure a more balanced assessment across different option contracts, we instead measure pricing errors using the discrepancy between the Black-Scholes implied volatility (BSIV) derived from model-predicted prices and that implied by observed market prices. Specifically, the absolute pricing error for contract i on date t is:

$$\widehat{\epsilon}_{it} = |\sigma(P_{it}; K_{it}, \tau_{it}, r_{it}, S_t, d_t) - \sigma(\widehat{P}_{it}; K_{it}, \tau_{it}, r_{it}, S_t, d_t)|, \tag{15}$$

where P_{it} denotes the market price (measured by the closing mid-quote on day t), and $\widehat{P}_{it}$ denotes the price predicted by the TL, DL, or Heston model. The function $\sigma(\cdot; K_{it}, T_{it}, r_{it}, S_t, d_t)$ maps the price for contract i on date t into the BSIV. The interest rate r_{it} is obtained by linearly interpolating the risk-free yield curve at the corresponding maturity of the contract. We compute σ_{it} using the stochastic gradient descent technique.

[13] We deliberately maintain a relatively simple neural network architecture, as our objective is to compare TL with DL rather than to optimize absolute predictive performance. In practice, more can be done to tune the hyperparameters for each method.

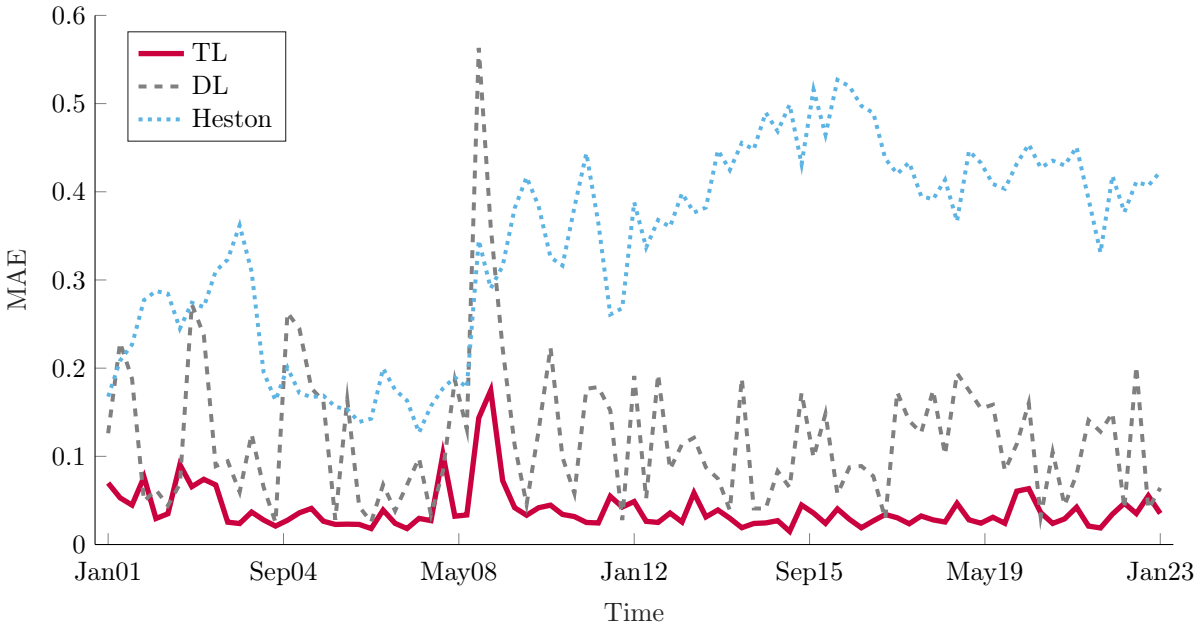

Figure 3: **Out-of-sample pricing errors.** This figure reports the out-of-sample median-absolute pricing errors (in terms of BSIV) for TL, DL, and the Heston model at quarterly frequency from 2001Q1 to 2023Q1. At the end of each quarter, the models are trained using data from the past 3 months, and the pricing errors are computed using data in the following 3 months.

For each three-month test set, we compute the median absolute BSIV-based pricing error (BSIV-MAE) across all contract-day observations. We use the median rather than the mean absolute error because model-predicted option prices may occasionally generate extreme outliers for the BSIVs or even preventing the computation of a valid BSIV altogether (when the predicted price falls outside the admissible range implied by the Black-Scholes model). Such extreme values can disproportionately affect the mean error, whereas the median is robust to outliers.

Figure 3 shows that the TL model consistently achieves lower BSIV-MAEs across all time periods compared with both the DL model and the Heston model. Over the full sample, the BSIV-MAE for the TL model is on average 0.0394, compared to 0.1215 for DL and 0.3403 for the Heston model, which correspond to error reduction of 67.6% and 90.8%, respectively. The relatively poor performance of the Heston model is unsurprising. Modern structural option pricing models have been shown to exhibit substantial parameter instability (see e.g., Chen, Didisheim, and Scheidegger, 2025), making it difficult for them to deliver

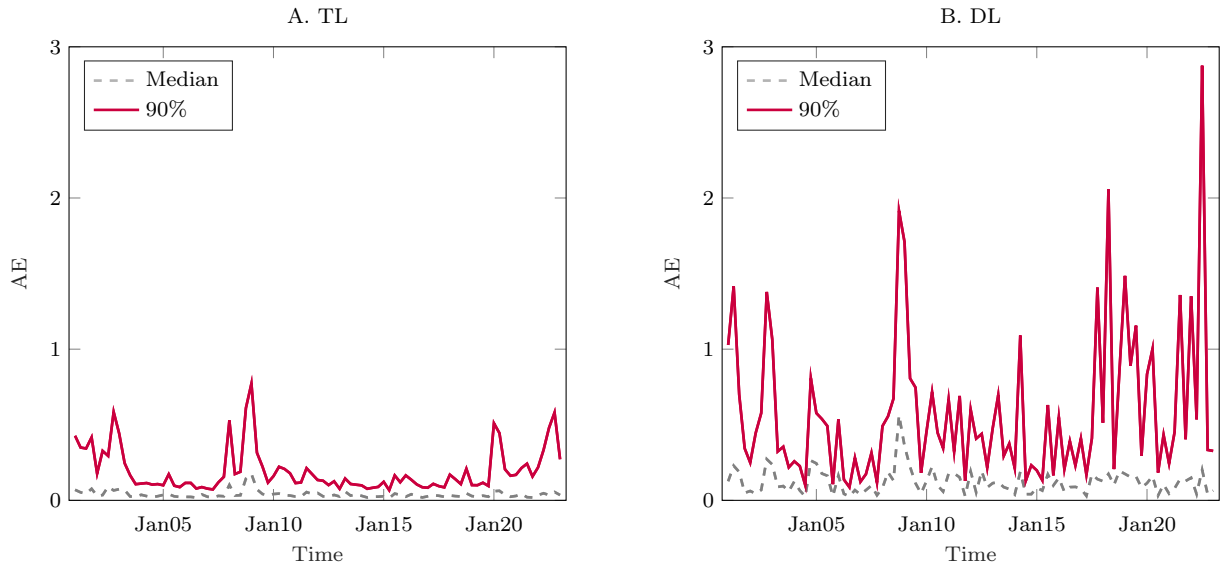

Figure 4: **Outliers for TL vs. DL.** In this figure, we compare the 90th percentiles of out-of-sample pricing errors (absolute errors in BSIV) for the TL and DL model.

accurate out-of-sample pricing performance. Over time, the performance of the Heston model deteriorates markedly after 2008, in part due to the growing share of short-dated options in the data. Although the DL model generally outperforms the Heston model over the full sample, its pricing errors display a pronounced spike during the 2008 financial crisis.

Performance Stability. In addition to achieving higher average accuracy, the TL model demonstrates significantly greater stability compared to the DL model. This enhanced stability is due to the theory-informed regularization reducing the variance of out-of-sample forecasts, which is particularly relevant for complex and data-constrained financial applications. Transfer learning improves the stability of deep learning in the following three ways: i) it compresses the tails of the prediction-error distribution; ii) it is less sensitive to limited sample sizes; and iii) its performance is less influenced by the inherent randomness in the neural network training process.

Regarding the first point about the tail behavior of the prediction-error distribution, we plot the 90th percentiles of pricing errors for both the TL and DL models in Figure 4. The results show that the TL model produces less extreme errors compared to the DL model. Over the full sample, the 90th percentile of out-of-sample absolute BSIV errors for the TL

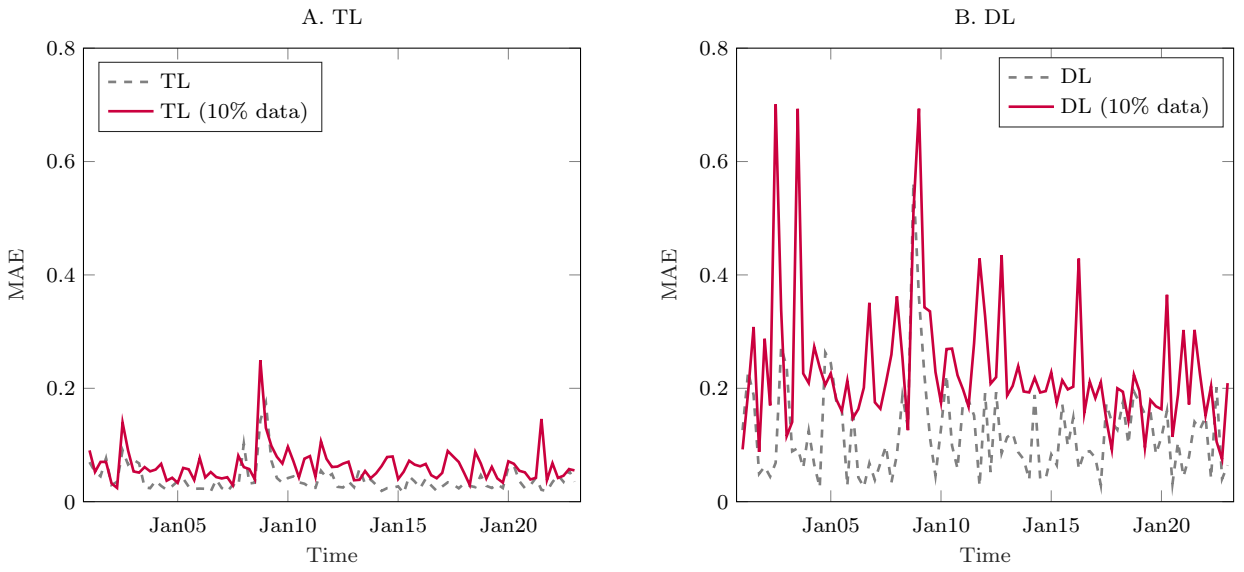

Figure 5: **TL vs. DL with small training sample.** In this figure, we compare the out-of-sample BSIV-MAE for TL against those of DL when the training sample is reduced to 10% of the original size.

model is on average 0.1972, compared to 0.5808 for the DL model, which is a reduction of 66%. This robustness in mitigating extreme prediction errors is particularly important in settings where the economic consequences of large pricing errors are severe.

Turning to the second point about model sensitivity to limited sample sizes, we plot in Figure 5 the BSIV-MAE for the TL and DL models when the training sample in each three-month period is reduced to 10% (randomly drawn) of its original size. Under this small-sample setting, the DL model experiences a substantial deterioration in performance, with the average BSIV-MAE increasing by 0.1125, and in some periods by considerably more (by 0.3508 at the 95th percentile), indicating a high sensitivity to data availability. In contrast, the TL model shows considerably more resilience, with the average BSIV-MAE only increasing by 0.0235 (or by 0.0554 at the 95th percentile).

This ability to learn effectively from limited data is particularly relevant for real-world financial and economic applications. It allows the TL model to deliver relatively stable and reliable predictions in environments with (effectively) sparse observations, such as with low-frequency macro data, newly introduced financial products with limited histories, or in settings in which the data-generating process is unstable due to structural breaks, for

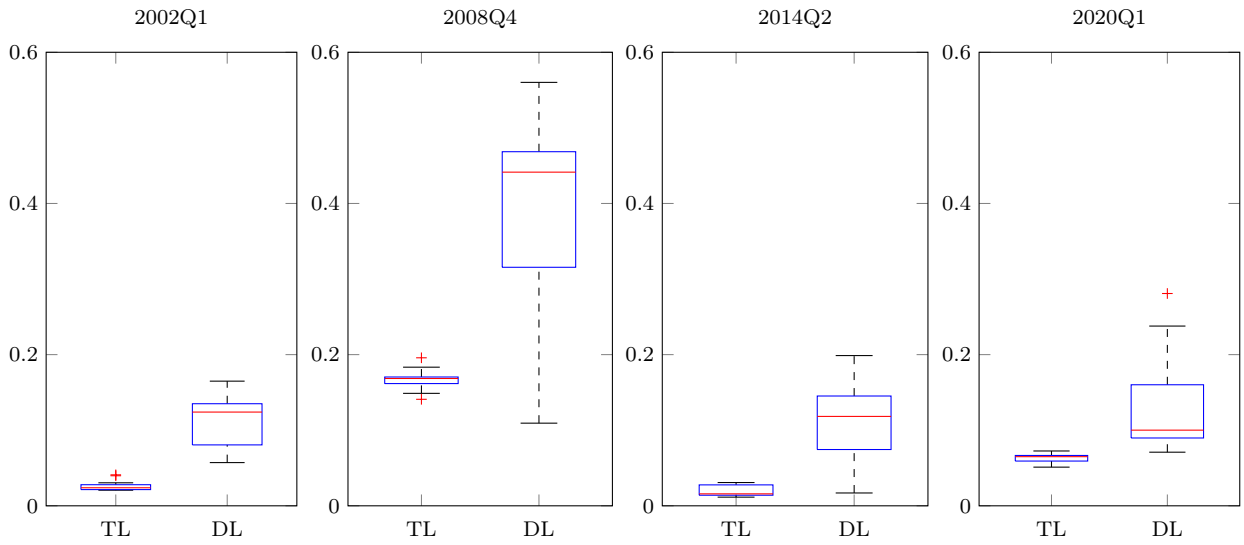

Figure 6: **Stability of TL vs. DL with respect to network training.** In this figure, we use boxplots to demonstrate the distribution of out-of-sample BSIV-MAE for the TL and DL models at four selected time points, as the random seeds used for network training vary. The box spans the first and third quartiles ($Q1$ and $Q3$), with its height representing the interquartile range, $IQR = Q3 - Q1$, and the horizontal line inside the box indicating the median. The lower and upper whiskers extend to the smallest and largest observations within $Q1 - 1.5 \times IQR$ and $Q3 + 1.5 \times IQR$, respectively. Observations outside this range are classified as outliers and plotted individually.

example following regulation changes.

Finally, we examine the third point about variation in model performance stemming from the inherent randomness of network training. Both the random initialization of network parameters and the mini-batch stochastic gradient descent algorithm contribute to the randomness of the trained model. To assess their impact on the TL and DL models, we retrain both models using different random seeds for four selected testing quarters. Two of these periods (2002Q1 and 2014Q2) correspond to relatively calm market conditions, while 2008Q4 coincides with the peak of the Great Recession and 2020Q1 marks the onset of the COVID-19 pandemic.

The results, reported in Figure 6, show that the TL model's pricing errors (BSIV-MAE) remain remarkably stable under different random seeds, indicating robustness to training randomness. In contrast, the DL model exhibits substantial sensitivity to changes in the random seed, especially during the crisis period of 2008Q4. Quantitatively, the inter-quartile

ranges of the BSIV-MAE for the DL model in the four quarters are 0.0513, 0.1353, 0.0574, and 0.0665, compared to 0.0060, 0.0071, 0.0126, and 0.0069 for the TL model. The superior stability of the TL model can again be attributed to the theory-informed regularization, which makes TL a more reliable choice in applications where consistency across training runs is critical.

At a fundamental level, the DL model relies on patterns extracted from historical data, implicitly assuming that future market behavior will resemble the past. As a result, its performance tends to deteriorate when training data are limited or noisy, as well as during periods of market turbulence. In contrast, the TL model takes advantage of the structural information derived from economic theory. Although such information can be biased (due to model misspecification), the results above demonstrate clear benefits from theory-informed variance reduction. We further examine this bias-variance trade-off in Section 4 as we study the sources of the TL model's superior performances.

4 Analyzing Transfer Learning's Performance

In this section, we dig deeper into the performances of TL versus DL models and examine a few potential explanations for the superior performances of the TL model.

4.1 When Does TL Perform Better?

We start by performing a relative performance attribution analysis for the TL vs. DL model. Specifically, for contract i on date t in the testing sample, we regress the differences in the absolute BSIV pricing errors defined in (15) between DL and TL on a variety of explanatory variables,[14] including contract characteristics and market conditions (see Table A.2 for detailed descriptions of the variables):

$$\widehat{\epsilon}_{it}^{DL} - \widehat{\epsilon}_{it}^{TL} = \alpha_t + x_{it}\beta + u_{it}. \tag{16}$$

[14]Observations for which $\widehat{\epsilon}_{it}$ are undefined are excluded from the regression.

Table 2: **Performance Attribution Analysis**

The table shows the results of the attribution analysis. The dependent variable is the difference in absolute pricing error (in terms of BSIV) between the DL and TL models for contract i on date t. The heteroskedasticity- and autocorrelation-consistent (HAC) standard errors are also reported. The total sample consists of 23,346,511 contract-day observations.

	(1)	(2)	(3)	(4)	(5)	(6)
0–7	-0.0217	-0.0216	-0.0214	-0.0214	-0.0213	-0.0236
	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
8–14	-0.0126	-0.0124	-0.0124	-0.0122	-0.0124	-0.0146
	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
90+	0.0022	0.0024	0.0026	0.0028	0.0022	0.0030
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
OTM	0.0349	0.0351	0.0343	0.0344	0.0356	0.0353
	0.0001	0.0001	0.0001	0.0001	0.0002	0.0002
DOTM	0.0785	0.0788	0.0777	0.0781	0.0775	0.0773
	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
ITM	0.0490	0.0487	0.0489	0.0486	0.0471	0.0478
	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
DITM	0.0275	0.0274	0.0276	0.0275	0.0264	0.0264
	0.0006	0.0006	0.0006	0.0006	0.0006	0.0006
BAspread			-0.4256	-0.4179	-0.6675	-0.6837
			0.0220	0.0218	0.0357	0.0364
VIX60					0.1167	0.0919
					0.0008	0.0008
Δ *VIX*					0.1814	0.1086
					0.0009	0.0009
abnvolume					0.0184	0.0202
					0.0003	0.0003
distance						1.9310
						0.0096
const	0.0532	0.0495	0.0539	0.0496	0.0346	0.0343
	0.0001	0.0002	0.0001	0.0002	0.0002	0.0002
Time FE	×	✓	×	✓	×	×
R^2	0.0070	0.0098	0.0074	0.0102	0.0121	0.0145

A positive coefficient in β indicates that an increase in the corresponding element of x_{it} is associated with a bigger performance advantage for the TL model.

Table 2 presents results. To examine how the relative performance of TL depends on option maturity, we include three dummy variables for maturity buckets: ultra-short (0-7

days), short (8-14 days), and long maturity (90+ days), with contracts of intermediate maturity (15-90 days) serving as the benchmark group. Across specifications, the coefficients on the 0-7 day and 8-14 day indicators are significantly negative at a 1% level, with the effect for ultra-short maturities being larger in magnitude. In contrast, the coefficient on the 90+ day indicator is generally significantly positive. These findings imply that the performance advantage of the TL model is smaller for short-dated options. As argued by Andersen, Fusari, and Todorov (2017), short-maturity option prices deviate more from traditional structural option pricing models, in part because they are particularly sensitive to jump risk and stochastic volatility dynamics in the underlying index. Since the Black-Scholes model does not take into account such features, it is more severely misspecified for short-dated options. This increased model bias helps explain the diminished advantage of the TL model in short-dated options.

Next, we examine how the relative performance of TL varies with option moneyness. Following Andersen, Fusari, and Todorov (2017), we measure moneyness as:

$$m = \frac{ln\left(\frac{K}{H_T}\right)}{\sqrt{T}IV_{atm,T}}, \tag{17}$$

where H_T represents the forward price with expiration T, and $IV_{atm,T}$ denotes the implied volatility of the option whose strike price is closest to H_T. Table A.2 provides the specific ranges for m we use to define DOTM (deep-out-of-the-money), OTM (out-of-the-money), ITM (in-the-money), and DITM (deep-in-the-money) options.

The coefficients on DOTM, OTM, ITM, and DITM are all significantly positive, with the largest magnitude for DOTM options. These results indicate that the performance gap between TL and DL is larger for both ITM and OTM options than for at-the-money (ATM) options, and is most pronounced for DOTM options. At first glance, this finding may appear surprising: because the Black-Scholes model cannot capture the volatility smile, it is more severely misspecified for OTM and ITM options, which should reduce the effectiveness of TL. However, relative to near-the-money options, OTM and ITM options have few observations (the adjacent strikes are further apart) and are generally less liquid. DOTM options, in particular, have considerably wider relative bid-ask spreads, making the mid-quote-based

option prices noisier. These factors are favorable for the TL model, which is better suited to learning in small-sample, high-noise environments.

In Columns (3) and (4), we add BAspread, the contract-day-level bid-ask spreads, as control. The coefficient estimate is significantly negative whether we include the time fixed effect or not, indicating that the performance advantage of TL relative to DL diminishes for less liquid contracts. Since the Black-Scholes model abstracts from liquidity frictions, its degree of misspecification is likely more severe for less liquid contracts. Thus, the increase in theory-induced bias can explain the result above.

In Column (5), we include three market-wide measures to examine how the relative performance of TL depends on the market conditions: i) VIX60, the past-60-day average of the VIX index; ii) ΔVIX, the difference between the VIX index on date t in the testing sample and the average VIX over the 3-month training-sample; iii) abnvolume, the cross-sectional average of 5-day moving average of abnormal volume over all option contracts, where daily abnormal volume is defined as contract i's trading volume on date t divided by the average of market trading volume over the previous three months.

The coefficient estimates for VIX60, ΔVIX, and abnvolume are all significantly positive. They suggest that the relative advantage of the TL model is more pronounced when the market (the level of the market index) is volatile, when the VIX index at the time of making the forecast is alleviated relative to the average VIX level observed during the training sample, and when the average trading volume in the market is high relative to the past three months.

As discussed earlier, the DL model effectively applies the patterns learned from the training data to the test set. The three variables above capture periods when market conditions differ substantially between the training and test samples, whether in terms of the level of the market index, the level of implied volatility, or the level of trading activities. Intuitively, when a model is trained using only data from a particular region of the feature space and is then asked to generate predictions in a different region, maintaining consistent performance in such settings requires strong out-of-distribution generalization, a task that standard deep learning models are known to struggle with.[15]

[15] A large body of theoretical and empirical work has shown that standard deep learning models excel at interpolation but struggle with extrapolation and out-of-distribution generalization, particularly under distribution shifts (see e.g., Zhang et al., 2017; Arjovsky et al., 2019; Geirhos et al., 2020).

In contrast, theory-guided transfer learning enhances out-of-distribution generalization by restricting the effective hypothesis space. Pretraining on the synthetic data from a structural model biases the learned representation toward an economically consistent mapping, reducing variance and limiting spurious extrapolation when fine-tuning on empirical data. As a result, the model extrapolates more stably beyond the support of the training distribution, mitigating the out-of-distribution failures typical of purely data-driven approaches.

To systematically compare the generalization capabilities of the TL and DL models, we use distance, the Mahalanobis distance between the input vector for contract i on date t, x_{it}, and the empirical distribution of the training data, to measure the degree to which a data point is "out of distribution":

$$d_M(x_{it}) = \sqrt{(x_{it} - \widehat{\mu}_t)\widehat{\Sigma}_t^{-1}(x_{it} - \widehat{\mu}_t)}, \tag{18}$$

where $\widehat{\mu}_t$ and $\widehat{\Sigma}_t$ are the estimated mean vector and covariance matrix of the input features in the corresponding training sample.

Column (6) of Table 2 shows that the coefficient estimate on the Mahalanobis distance is significantly positive, indicating that the relative performance advantage of TL over DL increases as test observations move farther away from the training distribution. This result corroborates the previous results based on market-wide measures and supports the hypothesis that theory-guided transfer learning enhances extrapolation beyond the support of the training data. Given that financial markets frequently undergo distributional shifts and structural breaks, such robustness to out-of-distribution inputs is particularly valuable in practice.

4.2 Is the Outperformance of TL Due to More Training Data?

Since the TL model is trained on both empirical and synthetic data, a natural question is whether the superior performance of the TL model relative to DL simply reflects its access to more training data. To address this concern, we consider three alternative ways to expand the training data for the DL model:

1. **Expanding-window DL**: In this approach, the training data for the DL model at the end of each quarter include all observations from the beginning of the sample (2000Q4)

up to the current quarter. As a result, the size of the training sample grows over time and eventually far exceeds the amount of empirical data used by the TL model or the rolling-window DL, which is always restricted to the most recent three months.

2. **Pooled DL**: In this approach, we align the training data for the DL and TL models by training DL on a dataset that combines empirical observations from the most recent three months with the synthetic observations used for source-domain training.

3. **Warm-start DL**: In this approach, the DL model is trained using a warm start, whereby the network parameters from the previous training cycle are used to initialize training on the current dataset. This approach effectively passes information from earlier data through the learned parameters.

Figure 7 presents the comparison results. As shown in Panel A, although the expanding-window DL model improves upon the rolling-window DL, it continues to underperform the TL model by a substantial margin. Over the full sample, the average BSIV-MAE for expanding-window DL is 0.0756, compared to 0.1215 for rolling-window DL and 0.0394 for TL. Notably, the performance gap between expanding-window DL and TL does not appear to narrow over time, despite the fact that the former is trained on an increasingly larger set of empirical observations than the latter. A key reason is that the underlying data-generating process likely evolves over time, reducing the relevance of older data. This result underscores the importance of effective learning from more recent but limited samples.

Panel B shows that the pooled DL model brings no meaningful improvement over the rolling-window DL, with an average BSIV-MAE of 0.1122 over the full sample. Although the data-pooling scheme nominally aligns the training data used by the TL and DL models, the two-step training procedure of TL effectively assigns endogenous weights to the synthetic and empirical data through theory-informed initialization and subsequent fine-tuning in the target domain. In contrast, under naive data mixing, synthetic observations dominate the pooled training sample (recall that the synthetic dataset has 800,000 observations), causing the pooled DL model to behave too much like the misspecified structural model. In Section 5.3, we further explore the relationship between TL and the optimal mixture of synthetic and empirical data within a Bayesian framework.

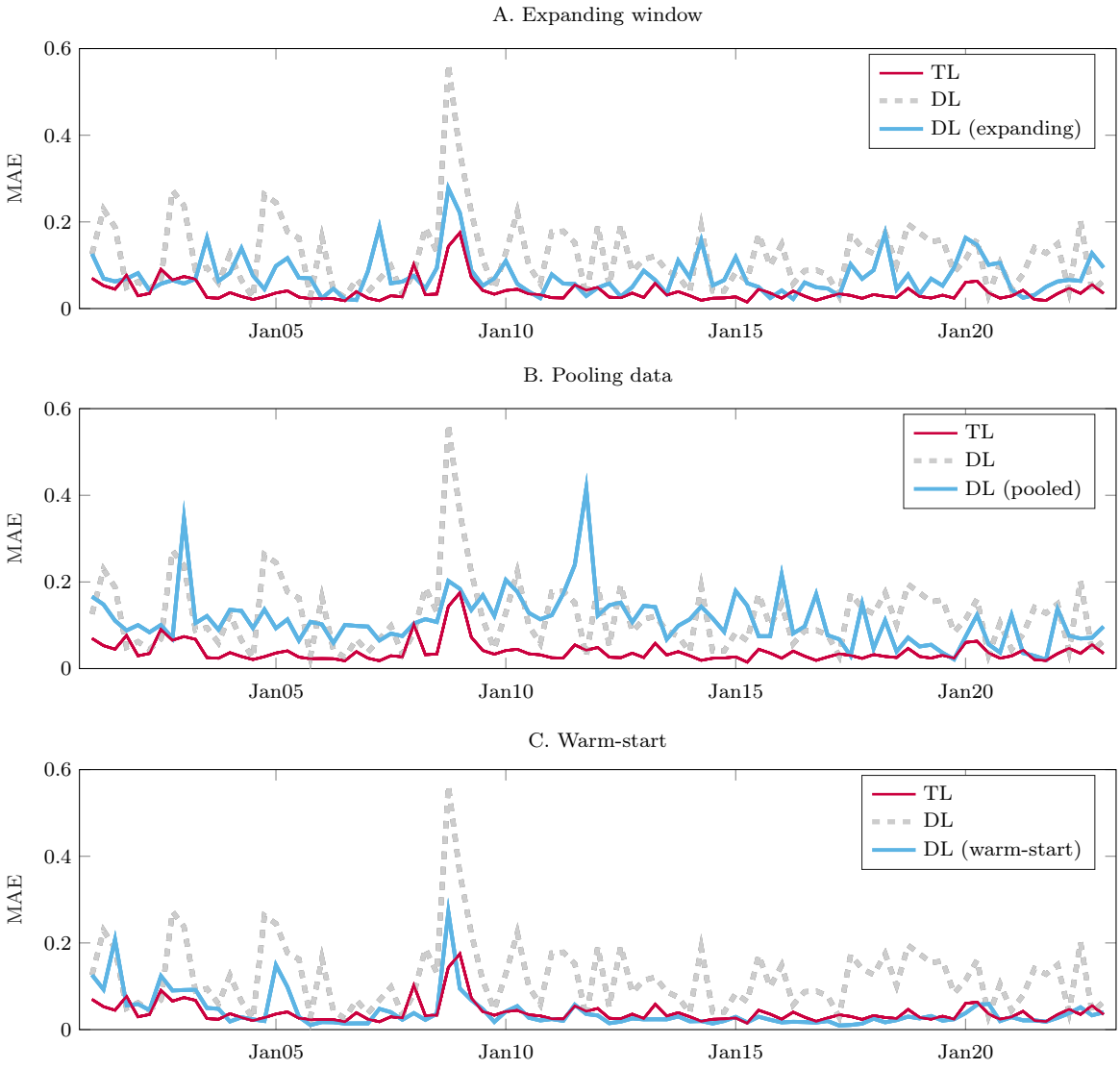

Figure 7: **TL vs. DL under expanding window, pooled data, and warm-starting.** This figure compares the out-of-sample pricing errors (BSIV-MAE) for TL and DL under the rolling-window setting against those of three differently-trained DLs: expanding-window DL, pooled DL, and warm-start DL.

Finally, Panel C shows that the warm-start DL brings the most significant performance improvement among the three alternative approaches. Over the full sample, the average BSIV-MAE for warm-start DL is 0.0411, which is substantially lower than that of both rolling-window and expanding-window DL, and only slightly higher than that of the TL model. It is also informative that a visible performance gap between warm-start DL and TL

persists during the first 8-10 years of the sample and only gradually narrows over time.

Like TL, the warm-start DL uses informed initialization when training on the most recent data, although its information is inherited from previous training cycles instead of from economic theory. The results above indicate that this inherited information is initially of low quality and improves gradually as additional empirical data accumulate. Combined with the same early stopping rule used for TL, warm-start DL thus achieves a form of informed regularization similar to that of TL. The eventual convergence in performance between warm-start DL and TL suggests that theory-informed regularization, despite misspecification of the structural model, can be as effective as that provided by substantially larger empirical datasets. In practice, however, additional empirical data may not always be available or may be subject to more pronounced distributional shifts than those observed in the option data. In such settings, theory-guided transfer learning will have a unique advantage over warm-start DL.

4.3 Is the Structural Model Simply Identifying Relevant Features?

Beyond generating synthetic data, economic theory may also improve machine learning performance by identifying economically relevant features. For example, while our TL and DL models use 16 input features, the Black-Scholes model relies only on 6 features. This raises the question of whether the superior performance of the TL model can be attributed to theory-guided feature selection, that is, whether the Black-Scholes model helps TL focus on a smaller set of relevant inputs and thereby reduces the risk of overfitting. To examine this hypothesis, we retrain both the TL and DL models using only the input features required by the Black-Scholes model.

Figure 8 reports the results. Panel A shows that restricting the input features to those used in the Black-Scholes model has little effect on TL performance: the average BSIV-MAE over the full sample increases only slightly, from 0.0394 to 0.0404. One notable exception is 2008Q4, where the BSIV-MAE rises from 0.1747 to 0.4384. In contrast, the DL model benefits substantially from feature restriction. Its average BSIV-MAE over the full sample declines by 42%, from 0.1215 to 0.0703, which is still well above that of the TL model. Finally, its performance in 2008Q4 also improves under the restricted-input specification.

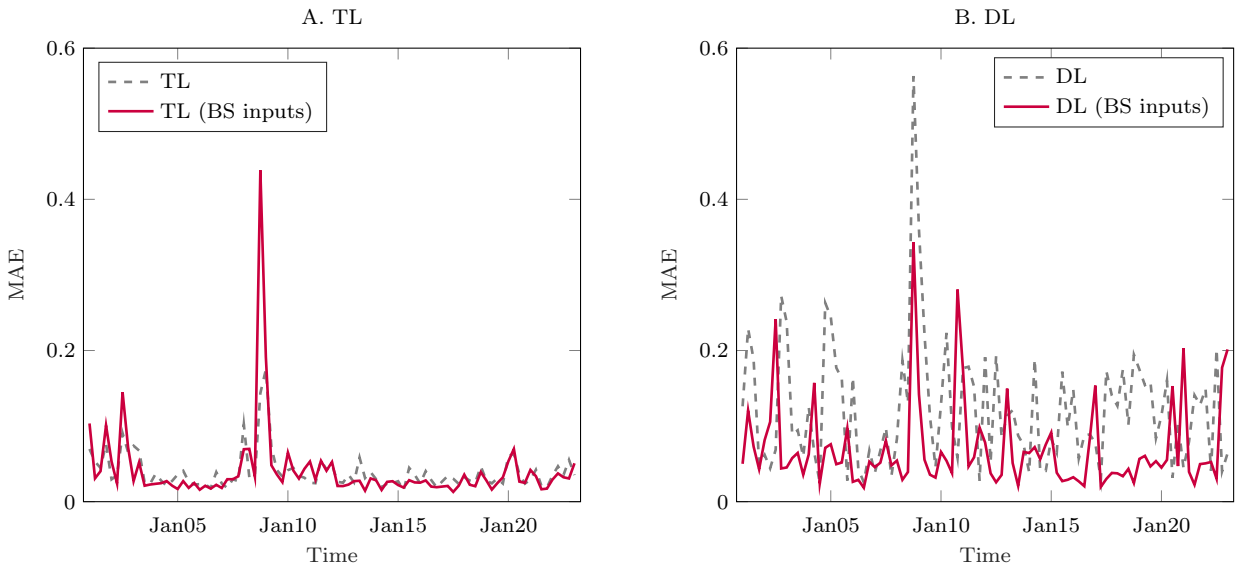

Figure 8: **TL and DL with Black-Scholes-implied inputs.** TL (BS inputs) refers to that within the transfer learning framework, we retain solely the inputs utilized in the Black-Scholes model. Similarly, DL (BS inputs) means that within the deep learning framework, we retain solely the inputs utilized in the Black-Scholes model.

The fact that the DL model continues to underperform the TL model even after restricting its inputs to those implied by the Black-Scholes model indicates that TL benefits from the structural model in ways that go beyond feature selection. In particular, the structural model likely provides additional information through the specific nonlinear relationships between option prices and inputs.

Moreover, the contrasting effects of feature restriction on the TL and DL models are also informative. When supplied with a large set of inputs, the DL model is prone to overfitting, and restricting its feature set helps mitigate this risk. In this case, the resulting reduction in variance outweighs the cost of discarding potentially informative features outside the Black-Scholes model, even though doing so may increase bias. Put differently, a standard deep learning model faces a dilemma: while incorporating additional features can bring more information, it also raises susceptibility to noisy inputs and overfitting. By contrast, the TL model does not suffer from the same degree of overfitting due to theory-informed regularization, which allows it to better exploit weaker predictors without being overwhelmed by noise. For example, features related to market liquidity (but outside the Black-Scholes

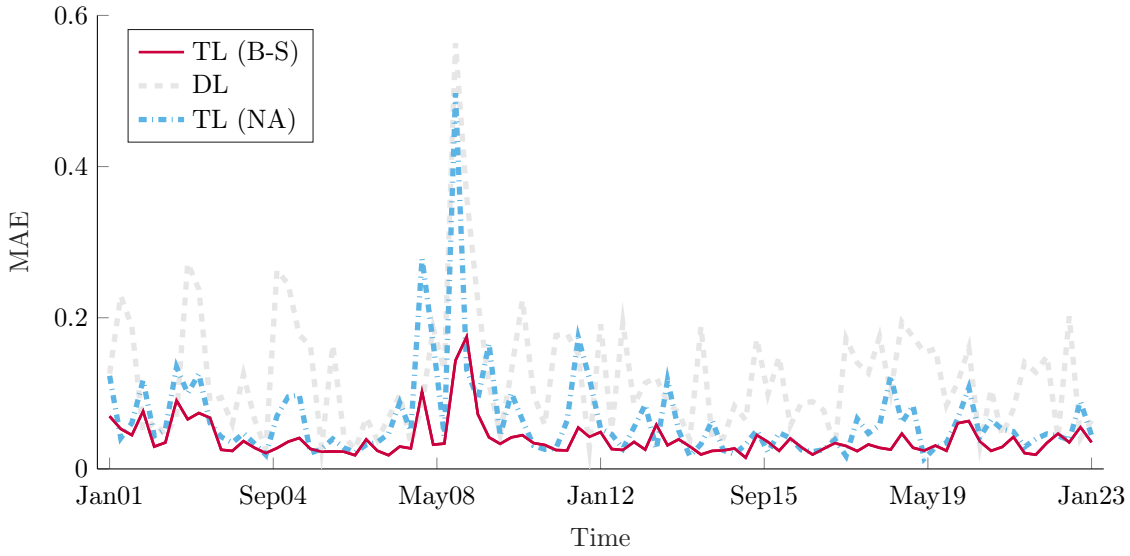

Figure 9: **TL with strong vs. weak structural restrictions.** In this figure, we compare the out-of-sample median-absolute pricing errors for TL that uses the Black-Scholes model in the source domain against the TL that uses the model-free no-arbitrage bounds in the source domain.

model) may contribute to the superior performance of TL during stress periods such as 2008Q4.

4.4 TL Performance under Different Structural Models

In Section 3.1, we have discussed the rationale for choosing the Black-Scholes model as the source-domain structural model in our application, including its absence of look-ahead bias and its suitability for illustrating how a misspecified model can nevertheless be useful within the TL framework. More broadly, we face a classical trade-off when choosing which economic model to use for the source domain: precision vs. robustness. One may select a model that delivers more precise restrictions but is also more prone to misspecification, or instead opt for a model that imposes weaker yet more robust restrictions on the data. Here, we examine this trade-off by training the TL model under arguably the most robust economic restriction, the model-free no-arbitrage bound.

Specifically, we consider the following no-arbitrage bounds for a European put option:

$$\max(Ke^{-rT} - Se^{-dT}, 0) \leqslant P \leqslant Ke^{-rT}. \tag{19}$$

To generate synthetic data according to this no-arbitrage bound, we first randomly draw the input features $\tilde{x}_i$ following the same procedure used in the main result, and then draw the option price $\tilde{y}_i$ from a uniform distribution over the range implied by the no-arbitrage bound in (19). The TL model is then trained using the pretraining and fine-tuning procedure.

The results, reported in Figure 9, indicate that economic restrictions derived from the no-arbitrage bound also lead to meaningful performance improvements under the TL framework. For the TL model trained using the no-arbitrage bound, the average BSIV-MAE over the full sample is 0.0661, representing a 46% reduction relative to the DL model (0.1215). At the same time, this error remains about 68% higher than that achieved by the TL model trained using the Black-Scholes model. In fact, the performance gap between the two TL specifications is consistent over time.

The comparison between the two TL models further highlights the bias-variance trade-off inherent in theory-guided transfer learning. While the no-arbitrage bound arguably introduces less bias than the Black-Scholes model, the restriction it imposes is also substantially weaker and therefore provides less effective regularization. This result suggests that, in searching for the optimal trade-off, it would be helpful to experiment with a range of economic restrictions or models on the precision-robustness spectrum. While severely misspecified restrictions may indeed result in negative transfer, as demonstrated in the simple example in Section 2.2, one should not shy away from structural models that provide stronger, albeit potentially misspecified, economic restrictions. The TL framework can benefit from the resulting variance reduction, while fine-tuning on empirical data provides a mechanism to mitigate the impact of misspecification.

4.5 Optimal Learning Rate

We have emphasized the importance of fine-tuning in the transfer learning framework. Using a small learning rate together with early stopping induced by a low patience parameter helps

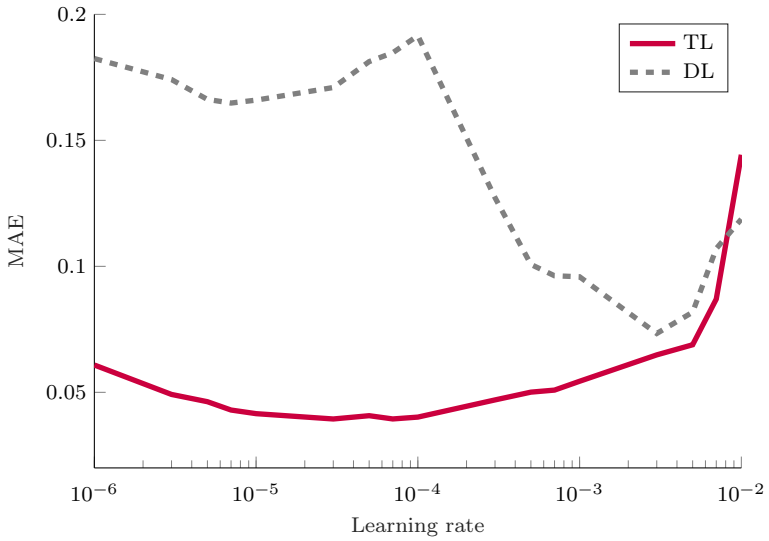

Figure 10: **Optimal Learning Rate in Target Domain.** The figure demonstrates the relationship between the learning rate and the out-of-sample pricing errors. We retrain the TL and DL model under different learning rates. The out-of-sample pricing errors (BSIV-MAE) are then averaged over the full sample.

reduce the risk of catastrophic forgetting, i.e., the loss of information acquired in the source domain. To illustrate this mechanism, Figure 10 shows the impact of different learning rates on the performances of the TL and DL models. The figure reveals a U-shaped relationship between model performance, measured by out-of-sample BSIV-MAE, and the learning rate for both models. Notably, the optimal learning rate for the TL model is approximately two orders of magnitude smaller than that for the DL model.

From the perspective of theory-informed regularization, a low learning rate helps to restrict the effective hypothesis space explored during target-domain fine-tuning. It makes it more difficult, although not impossible, for the network parameters to move far away a local neighborhood of the source-domain solution, thus enabling gradual adaptation to empirical data while preserving the structural information from economic theory.

5 Alternative Ways to Bring in Economic Information

In this section, we compare theory-guided transfer learning with other approaches for incorporating structural information into machine learning models. They include i) imposing

economic restrictions directly in the loss function, ii) boosting a structural model with a reduced-form model, and iii) Bayesian method.

5.1 Economics-informed Loss Function

If we believe that a machine learning model should satisfy certain structural restrictions implied by an economic model, we can directly impose such restrictions in the loss function such that any deviation would be penalized. This is exactly the approach taken by physics-informed neural networks (Raissi, Perdikaris, and Karniadakis, 2019). Following this approach, we can also build economics-informed loss function for the option pricing application,[16] where we directly penalize deviations from the Black-Scholes model in the loss function:

$$\mathcal{L}_{total} = \mathcal{L}_0 + \lambda_{BS}\mathcal{L}_{BS}, \tag{20}$$

where $\mathcal{L}_0$ corresponds to the loss on empirical data, while $\mathcal{L}_{BS}$ penalizes violation of the pricing restrictions from the Black-Scholes model, with $\lambda_{BS} \geqslant 0$ determining the relative weight on the economic restrictions.

It is worth noting that the new penalty term in the loss function (20) can also be viewed as a form of explicit regularization. As discussed in the target domain training step in Section 2.1, one may train the neural network on empirical data while penalizing deviations of the model parameters from their source-domain values. The term $\mathcal{L}_{BS}$ is effectively a specific implementation of the general regularization term $\mathcal{R}(\widehat{W}; \widetilde{W}^*)$ in (8), where the "distance" between $\widehat{W}$ and $\widetilde{W}^*$ is based on deviations of model predictions from the Black-Scholes-implied option prices.

A key challenge in implementing the loss function in (20) is the choice of the tuning parameter λ_{BS}. Unlike in physics-informed neural networks, economic models are more prone to misspecification. As a result, choosing a large value of λ_{BS}, which forces the machine learning model to adhere more closely to the economic restrictions, could introduce significant biases into the trained network. The optimal value of λ_{BS} thus depends on the trade-off

[16] To avoid the confusion with other ways to incorporate economic information into a neural network, including our transfer learning framework, we refer to the approach of imposing economic restrictions in the loss function as "economics-informed loss function" rather than "economics-informed neural networks".

between the degree of misspecification of the economic restrictions and the variance reduction they provide. For example, this trade-off is likely different if we replace the Black-Scholes restrictions with the no-arbitrage bound in (19), a more robust but less precise economic restriction.

For illustration, we use the loss function in (20) to train a DL model, setting λ_{BS} in each training episode based on the ratio of the sample sizes of synthetic and empirical data used in training. Besides avoiding costly tuning steps, this choice facilitates a comparison with the pooled DL model discussed in Section 4.2, which trains the DL model using the combined synthetic and empirical dataset.

The results are shown in Panel A of Figure 11. On the one hand, the DL model trained under the Black-Scholes-augmented loss performs consistently worse out of sample than the TL model, with an average BSIV-MAE over the full sample is 0.0977, compared with 0.0394 for TL. On the other hand, its performance exceeds that of both the standard DL and the pooled DL (see Figure 7). This naturally raises the question of whether there exists a feasible ex-ante strategy for tuning the parameter λ_{BS} such that the DL model with the economics-informed loss function can potentially match or outperform the TL model. We return to this question in Section 5.3.

5.2 Boosting Method

The second alternative approach we consider is inspired by Almeida et al. (2023), who combine economic model with machine learning through a boosting method, that is, using a reduced-form machine learning model to correct the errors of an economic model. Specifically, we first calculate the pricing errors for the Black-Scholes model in the empirical data. We then run an OLS regression of the pricing errors on the same 16 input features employed by the TL and DL models. The fitted values from this regression are subsequently added to the Black-Scholes prediction to obtain the final predicted option price:

$$\widehat{y}_i = BS(x_i) + x_i'\widehat{\beta}. \tag{21}$$

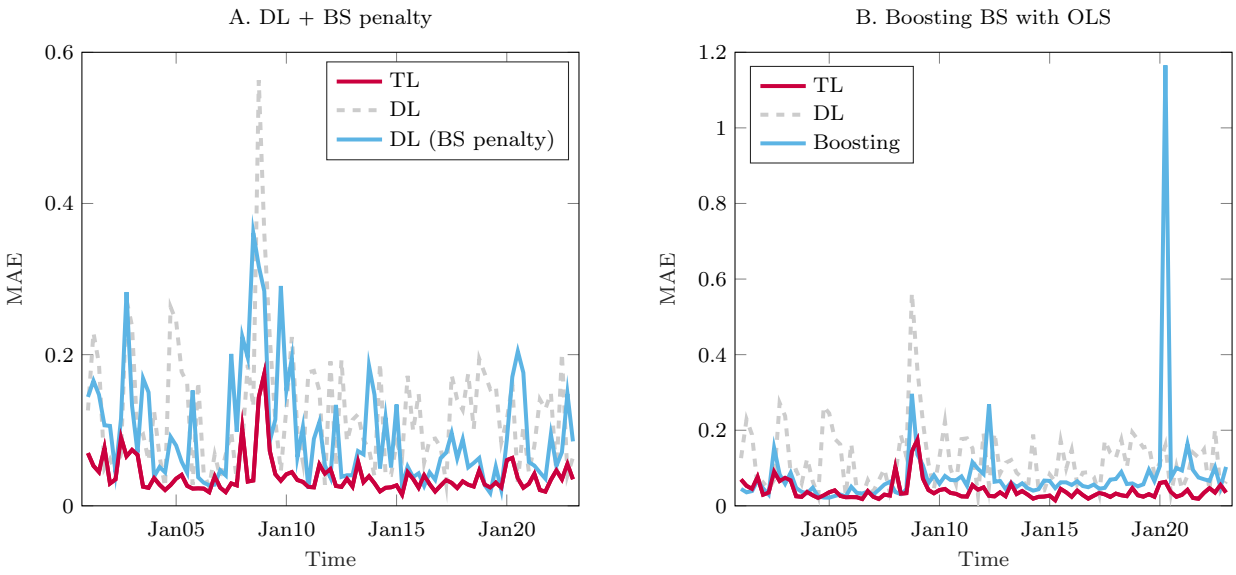

Figure 11: **TL vs. economics-informed loss and boosting.** In this figure, we compare the out-of-sample BSIV-MAE for TL against those of DL DL model trained with the Black-Scholes-augmented loss function (Panel A) and a Black-Scholes boosting model (Panel B).

We train this boosted model using the same rolling-window procedure, re-estimating the regression coefficients β in each three-month training period.

The results for this boosting method are reported in Panel B of Figure 11. Over the full sample, the average BSIV-MAE for the boosting model is 0.0798, approximately twice that of the TL model, but about two-thirds of that of the DL model. Moreover, the boosting model occasionally exhibits substantial pricing errors. For example, in 2020Q2, the BSIV-MAE generated by the boosting approach exceeds that of the TL model by more than a factor of 18, indicating substantial instability. This instability likely arises because the residual-learning procedure amplifies the noise-to-signal ratio in the target variables, making the subsequent training of the machine learning component (whether linear or nonlinear) even more vulnerable to over-fitting. This issue is particularly severe during periods when Black-Scholes pricing errors become large and highly volatile over time.

5.3 Bayesian Method

The use of simulated data from a structural model to enhance predictive performance has been explored in the context of Bayesian methods. For example, Del Negro and Schorfheide

(2004) apply a Bayesian vector autoregression framework to macroeconomic forecasting, where the prior for the VAR parameters is derived from a DSGE model. Effectively, one simulates data from the DSGE model, derives a prior for the VAR parameters by fitting the VAR to the simulated data, and finally computes the posterior distribution of the VAR parameters using the prior and the likelihood of empirical data. In the Bayesian setting, the ratio of the sample sizes of simulated to empirical data is a crucial hyperparameter. The more simulated data one uses relative to empirical data, the more weight is given to the theoretical model. Consequently, the relationship between prediction errors and the ratio of simulated to empirical data often exhibits a U-shaped pattern: either too much or too little simulated data can lead to suboptimal predictions (see e.g., Del Negro and Schorfheide, 2004).

Our transfer learning framework offers several advantages over the Bayesian-VAR approach, including: i) ability to capture rich nonlinear relationships; ii) ease of implementation, as it avoids Bayesian updating in high-dimensional parameter spaces; and iii) avoiding an explicit search for the optimal mixture between synthetic and empirical data. The last point is also an advantage of TL over the approach of economics-informed loss function discussed in Section 5.1, where one needs to tune the weights on the economic restrictions.

In the TL framework, training proceeds sequentially: first in the source domain using synthetic data, and subsequently in the target domain using empirical data. Because pretraining is confined to the source domain, increasing the size of the synthetic dataset improves the accuracy with which the source-domain model captures the underlying economic restrictions, but does not dilute the informational value of the empirical data to the final model. The relative influence of the theoretical model and the empirical data is primarily determined by the fine-tuning process in the target domain. In particular, a smaller learning rate combined with a lower patience parameter for early stopping makes it less likely for the network weights to deviate substantially from their pretrained values, thereby effectively assigning more weight to the theoretical model. Conversely, a larger learning rate and a higher patience parameter will amount to giving more weight to the empirical data.

To verify this intuition, we select four testing quarters to examine how varying the ratio of the sample sizes of synthetic and empirical data, denoted by α, affects the relative performance of TL vs. DL. The four quarters are the same as those used in Figure 6: 2002Q1, 2008Q4,

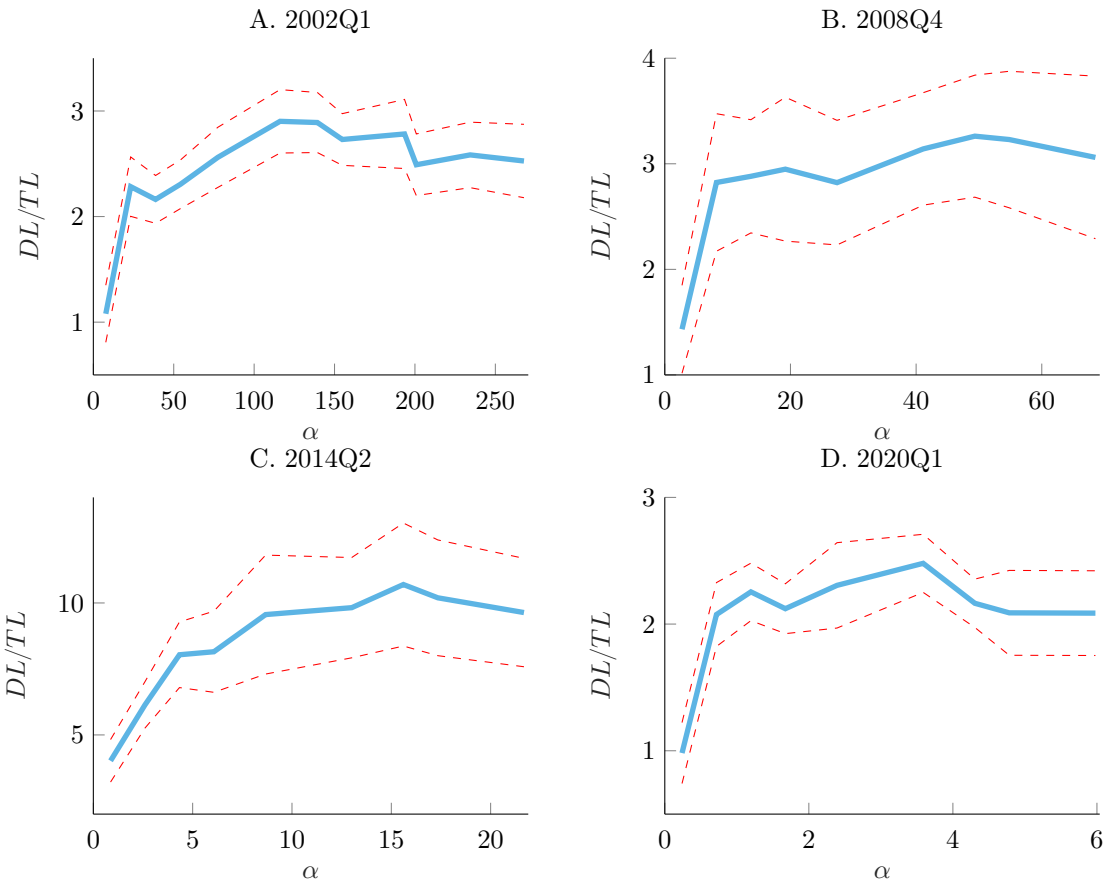

Figure 12: **Relative sample size of synthetic data and TL performance.** This figure demonstrates the effect of changing the sample size ratio α between synthetic data and empirical data on the relative performance of TL vs. DL. The y variable is the BSIV-MAE of DL divided by that of TL for a given testing quarter.

2014Q2, and 2020Q1. The results are presented in Figure 12. The relative performance of TL over DL is measured by the inverse of the ratio of their BSIV-MAEs. As α increases, the relative performance of TL generally improves, due to the fact that theory-informed initial weights become less noisy with more synthetic data. While this benefit diminishes as α grows large, there is no evidence that large values of α lead to worse performance, in contrast to the U-shaped pattern observed in Bayesian methods. This finding suggests that, in the TL framework, when determining the size of the synthetic dataset for pretraining, a simple and effective strategy is to use a sufficiently large synthetic sample.

How does the TL model perform relative to a Bayesian method under the optimal relative sample size α? Directly implementing a Bayesian method is infeasible in our setting due

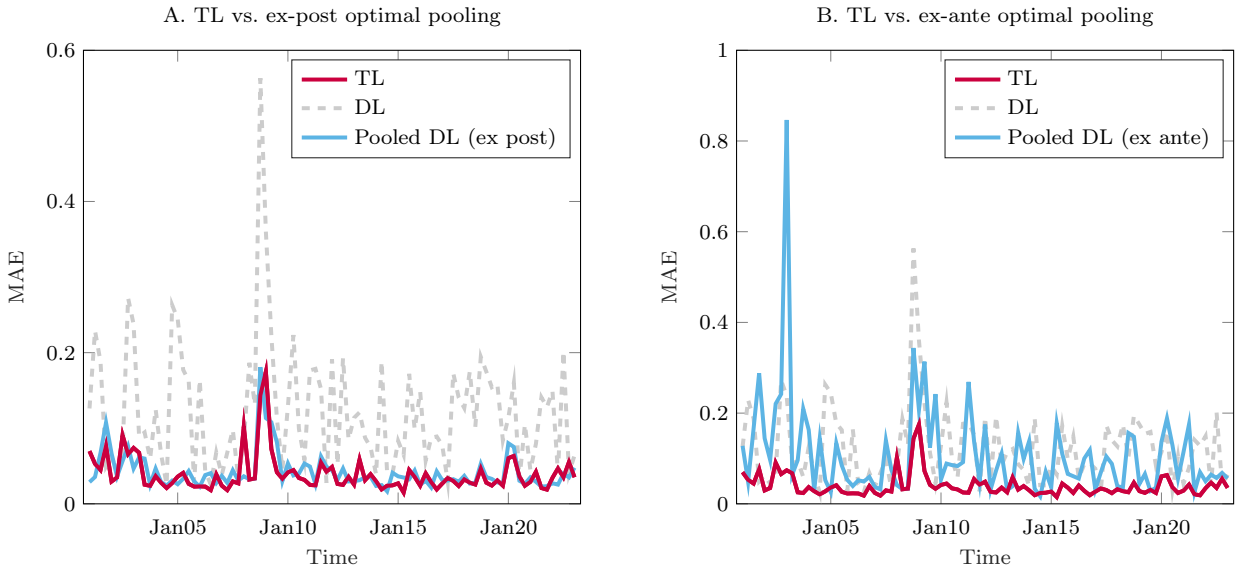

Figure 13: **Optimal Performance of Bayesian Method.** This figure presents the error curves achieved under the posterior optimal approximation using a Bayesian method, compared with those obtained from transfer learning and deep learning. At each point, we train multiple weights with synthetic and real data, then select the weight that performs best on the test set.

to the high dimensionality of the neural network parameter space. Instead, we mimic the Bayesian approach by training a DL model on a pooled dataset that combines synthetic and empirical data (i.e., pooled DL), where the mixing ratio between the two data sources is tuned either ex post or ex ante.

Under ex post tuning, we train the pooled DL model over a grid of relative sample size values, with $\alpha/(1+\alpha)$ ranging from 0 to 1, and select the value that minimizes the BSIV-MAE on the *test set*. This procedure is obviously not feasible in practice since it relies on future information. Nevertheless, it provides an upper bound on achievable performance under an optimally chosen weighting between synthetic and empirical data. Under ex ante tuning, we avoid using future information by setting the relative sample size for the current testing quarter equal to the optimal ex-post relative sample size obtained in the previous testing quarter.

Panel A of Figure 13 shows that TL performance is very close to that of the pooled DL under the *ex post* optimal weighting of sample size (which has an average BSIV-MAE over the full sample of 0.0420, vs. 0.0394 for TL), despite the fact that the latter uses future

information. Although this result does not imply that TL always recovers the optimal weighting between synthetic and empirical data – the results likely depend on the choice of target-domain hyperparameters including the learning rate and patience parameter, it suggests that TL has the capacity to approximate the Bayesian benchmark under optimal weighting implicitly through its fine-tuning process.

In contrast, Panel B of Figure 13 shows that the performance of the pooled DL under the *ex ante* optimal weighting (with an average BSIV-MAE of 0.1085) performs substantially worse than the TL model. This gap reflects the instability of the optimal weighting of sample size from one testing quarter to the next, which could stem from time variation in the degree of misspecification of the Black-Scholes model. During periods when the fit of the Black-Scholes model deteriorates, the optimal weighting should, all else equal, shift towards empirical data, and vice versa. Such shifts, however, are difficult to anticipate ex ante, making it challenging to tune the weighting in practice.

6 What Does TL Teach Us about Economics?

We have so far focused on how economic theory can enhance machine learning models through transfer learning. In this section, we turn to the converse question: how the transfer learning framework can inform and potentially improve economic theory.

One natural avenue is to conduct a relative performance analysis between the TL model and the structural model, analogous to the comparison between TL and DL in Section 4.1. Such an analysis can help identify specific market conditions or option characteristics, such as short maturities, extreme moneyness, high volatility or low liquidity regimes, under which the structural model performs poorly relative to TL, thereby shedding light on the limitations of existing theories. In addition, interpretability tools such as accumulated local effects (ALE) can be used to uncover nonlinear functional relationships exploited by the TL model that are not captured by the structural model.

Below, we examine two additional perspectives. First, through feature importance analysis, we can search for promising directions to extend existing theories. Second, the TL framework provides a new metric for model comparison, based on how much a structural

model complements the empirical data rather than how well it fits them.

6.1 Feature Importance Analysis

By comparing the feature importance of both the structural model inputs and the additional inputs introduced in the TL model, we can identify informative features that are absent from the structural model. One can then try to extend the structural model to incorporate such features to improve its out-of-sample (predictive) performance. Given the ease of incorporating new features into the TL framework, this approach provides a systematic and efficient way to search for potential directions to improve existing theories.

We measure feature importance using mean imputation: it is based on the incremental rise in the loss function when a given feature is effectively removed from the TL model. Specifically, let the baseline loss function in the test set be denoted by $\widehat{\mathcal{L}}_{\text{baseline}}$. For the jth feature, we replace its values with its sample mean in the training data ($x_{ij} = \overline{x}_{\cdot j}$ for all i), holding all other features fixed. We then compute the new loss in the test set, $\widehat{\mathcal{L}}_{-j}$. The importance of the jth feature is defined as the difference $\widehat{\mathcal{L}}_{-j} - \widehat{\mathcal{L}}_{\text{baseline}}$. Compared with alternative approaches such as SHAP-value-based methods (Shapley, 1953; Lundberg and Lee, 2017), this approach is computationally efficient, though it does not account for interactions among features.

We calculate this measure for the 16 features in each 3-month testing window and present the results using boxplots in Figure 14. The top panel displays the feature importance for the six inputs used in the Black-Scholes model, while the bottom panel shows the importance for the ten additional features introduced in the TL model.

The three most important features are all from within the Black-Scholes model: the strike price, the underlying stock price, and the risk-free rate. Their average feature-importance scores are substantially larger than those of the remaining variables. One subtle point is that the interest rate is computed at the contract level by linearly interpolating the risk-free yield curve at the corresponding maturity. As a result, r_{it} captures not only information about the level of risk-free rates but also maturity information across contracts, which may explain the relatively low feature importance assigned to time to maturity, τ_{it}.

Following the top three features, the next most important feature is the one-day-lagged VIX

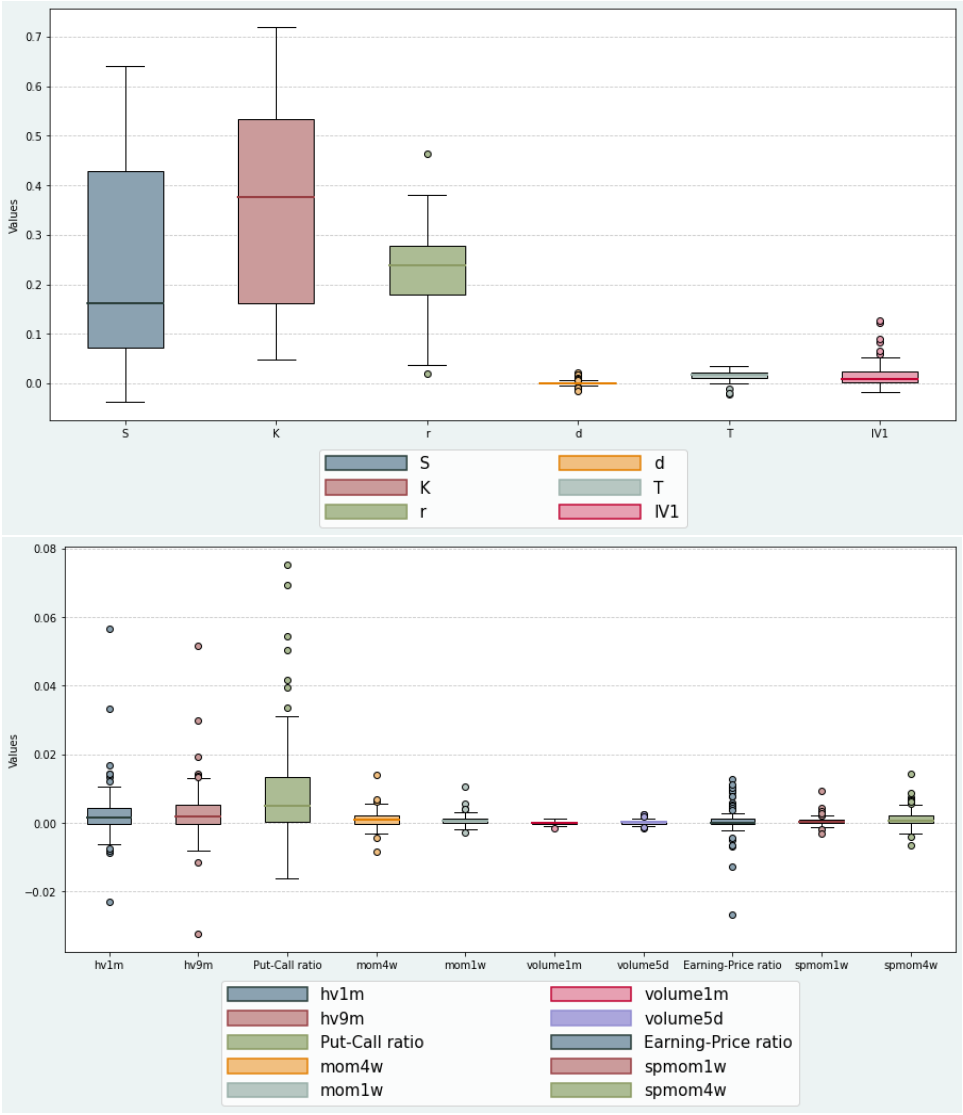

Figure 14: **Feature Importance.** The figure below shows a boxplot of feature importance for a transfer learning model. We have calculated the feature importance of the same feature in each quarter. This figure aims to visualize the comprehensive features of feature importance on different training sets. The outlier cut-off points of the boxplots were set to Q1-1.5IQR and Q3+1.5IQR. Data falling outside the cutoff point for outliers are marked separately.

($IV1$), which serves as our empirical proxy for the volatility parameter in the Black-Scholes model. Although its average feature-importance score is lower, $IV1$ exhibits markedly higher importance in certain quarters. As before, one explanation for $IV1$'s relatively modest average importance is that part of its informational content overlaps with other volatility-related features, such as the two historical volatility measures, $hv1m$ and $hv9m$.

Next, among the features outside of the Black-Scholes model, the put-call ratio (PCratio) stands out as the most important and distinct relative to those in the Black-Scholes model, with importance scores comparable to those of $IV1$. The put-call ratio, computed as the ratio of total open interest in SPX put options to that in SPX call options, likely reflects market sentiment and hedging demand, for example, for downside risks. Although such quantity-based information plays no role in classical no-arbitrage option pricing theories, they are relevant in models that feature constrained intermediaries and demand (Bollen and Whaley, 2004; Gârleanu, Pedersen, and Poteshman, 2009) or supply shocks (Chen, Joslin, and Ni, 2019). Its relevance in the TL framework suggests that incorporating such intermediary frictions into a quantitative option pricing model could meaningfully improve their empirical performances. Similarly, the relevance of $hv1m$ and $hv9m$ indicates that historical volatility measures provide useful information beyond implied volatility about market's expectations of future volatility dynamics.

At the same time, identifying features that are theoretically appealing yet exhibit low importance in the TL model can also provide guidance for how (not to) advance existing theories. For example, we include $volume5d$ and $volume1m$, the 5-day and 1-month moving averages of daily abnormal trading volume for a contract, as proxies for option liquidity. Somewhat surprisingly, their feature-importance scores are among the lowest, despite the common view that liquidity plays an important role in option pricing, especially for thinly traded contracts. One possible explanation is that index options are less sensitive to liquidity frictions than individual equity options. Of course, it is also possible that these volume-based measures may not adequately capture the liquidity effects.

In summary, feature importance analysis can help identify promising features that are absent from existing theories, thereby guiding future theoretical developments. It is worth emphasizing that the choice of loss function used to compute feature importance can help steering theoretical advancements in different directions. For example, while we compute feature importance based on the TL model's out-of-sample pricing performance, alternative loss functions can be employed when the objective is to improve other dimensions of the theoretical model, such as explaining option returns or hedging effectiveness.

6.2 Model Comparison

The TL framework provides a new metric for comparing structural models. Traditional model comparison often relies on metrics such as in-sample fit (e.g., based on the maximized likelihood or the GMM J-statistic) or out-of-sample predictive accuracy. These methods assess a model's performance in isolation, taking its restrictions and predictions at face value. These approaches evaluate a model in isolation, taking its implied restrictions at face value. In contrast, the TL framework allows us to compare structural models by the extent to which they complement empirical data in improving predictive performance. This is because, within this framework, structural restrictions are treated as an "informative prior" that guides and regularizes the data-driven learning process.

For example, consider two structural models, A and B. When evaluated in isolation, model A may fit historical data better or generate more accurate out-of-sample predictions than model B, making it the preferred choice under traditional model comparison criteria. However, it is possible that using model B as the source-domain model leads to a higher-performing TL model than using model A. This outcome can arise if the structural restrictions implied by model A are relatively easy to learn directly from empirical data, whereas those implied by model B are more difficult to recover. In such cases, provided that it is generally aligned with the true data-generating process (i.e., the degree of misspecification is not too severe), the information provided by model B could be more complementary to that already embedded in the empirical data and therefore provide more effective regularization within the TL framework. This new perspective of model comparison based on model-data-complementarity can be particularly relevant in the era of big data and AI.

7 Conclusion

In this paper, we develop a theory-guided transfer learning framework that uses structured restrictions from economic models to regularize the training of reduced-form machine learning models. Empirical results from the option-pricing application show that this framework delivers substantial improvements in out-of-sample predictive performance by leveraging

information from the classical Black-Scholes model. Attribution analysis further indicates that these gains are most pronounced in environments characterized by limited data, high noise, and unstable market conditions. While our implementation focuses on neural networks, the framework applies more broadly to a wide class of parametric machine learning models.

Beyond performance, the TL framework offers a new lens through which to evaluate and improve structural models. Feature-importance and relative performance analyses can provide systematic diagnostics for locating dimensions in which existing theories perform poorly, uncovering important missing features, and identifying promising directions to extend existing theories. Moreover, the TL framework introduces a new criterion for model comparison based on a structural model's complementarity with empirical data rather than its standalone fit.

Ultimately, our study suggests that economic theories can and should play a significant role in the age of artificial intelligence. Especially in settings where data are scarce, noisy, non-stationary, or subject to structural breaks, the theory-guided transfer learning framework provides an effective way to utilize potentially misspecified structural models to guide our learning from the data. It resonates with Box's observation: *"All models are wrong, but some are useful."*

Appendix

A Details of Neural Network Architecture and Training

The employed neural architecture in this article encompasses 16 hidden layers with 16 input variables each. Each hidden layer consists of 22 neurons and employs Leaky ReLU as the activation function. These hidden layers are linear transformations, constituting fully connected layers. To address the gradient vanishing problem, we introduced the ResNet-style structure. Detailed below are the specific parameters and structures utilized for all our results.

Residual Learning and Skip Connect We use the residual learning method to avoid the Vanishing Gradient Problem (VGP). The method allows us to make the network deep enough to implement transfer learning algorithms. In the application of deep learning to asset pricing, a natural question is, do deeper networks generalize better? In the research of computer science, the answer to the above question is no. The most immediate problem is the phenomenon of vanishing gradients and exploding gradients: the training of neural networks relies on backpropagation. If the depth of the neural network is too large, the neurons in the earlier layers of the neural network may obtain gradients close to 0 or abnormally large in backpropagation. As revealed by experiments in He and Sun (2015) and shown in Srivastava, Greff, and Schmidhuber (2015), when the network depth is too deep, the model performance will decrease with the depth of the neural network, which is called the degradation problem. The degradation problem is not caused by overfitting but by the structure of the neural network itself and the characteristics of the training method.

It should be noted that some AI tasks of asset pricing need to fit highly nonlinear equations, such as the fitting of option pricing equations that should be performed in this paper. This means that we need a deeper network with higher expressive power, so the degradation problem of the neural networks is an important problem to be solved. He, Zhang, Ren, and Sun (2016a) proposed a method called the residual learning method to solve the degradation problem. Their network structure allows neural networks to enjoy the benefits of

out-of-sample fitting capabilities produced by deeper networks without facing the problems of deep networks. The core idea of this method is to add a never-closed Shortcut Connections to the neural network, which is equivalent to learning residuals.

Mathematically, let a represent the input to a residual block, and $G(a)$ denotes the output from a series of operations applied to a, potentially including processes like convolution, activation, and normalization. A basic residual block can then be reformulated as:

$$b = G(a, \{V_j\}) + a \tag{A.1}$$

Here, b is the output of the residual block, $G(a, \{V_j\})$ signifies the transformation applied to the input a, with $\{V_j\}$ representing the set of parameters used in the transformation. The a term serves as a "shortcut" or "skip" connection, which is directly added to the transformation's output. The advantage of this structure is that even in very deep networks, residual connections help direct the gradient flow to earlier layers, thus improving the training performance of deep networks. We have drawn inspiration from the concepts of ResNet-style structure (He, Zhang, Ren, and Sun, 2016a,b), but unlike prior applications in computer vision, we have adapted these ideas to suit the unique characteristics of our task by incorporating skip connections into our deep network. The specific structure can be found in Figure 2. Detailed network parameters are discussed below.

Transfer Learning: For the source domain, we generate a series of random factor values and computed the prices produced by the Black-Scholes model for each data point. Employing these factor values as input variables and the prices derived from the Black-Scholes model as output variables, we conduct training using a total of 800,000 data instances. Training encompasses 200 epochs, with a learning rate of 0.0002 and 600 batches per epoch. We compute three distinct loss functions: an option Delta-weighted Mean Absolute Error (MAE) of predicted prices, MAE of predicted option Delta, and MAE of predicted option Vega. To prioritize minimizing pricing errors, we set the weights on the three loss terms, $\lambda_0, \lambda_1, \lambda_2$, at $10 : 2 : 2$. See Appendix C for robustness checks with different weight combinations.

For the target domain, we fine-tuned the model using real data's factor values as input

variables and actual prices as output variables. The training and validation sets together covered 3 months of data, with a 8:2 ratio between them, while the testing set consisted of an additional 3 months of data. The fine-tuning stage employed an early stopping strategy, terminating training early if the validation loss increased consecutively over two epochs. This early stopping criterion is applied to all models discussed below. The learning rate is set to 0.0001, which is determined through a sparse grid search on data preceding the first train-test pair. The batch size is calculated by dividing the training set sample size by 600. Backpropagation utilized the weighted MAE of predicted prices and actual prices with Delta as weights as the loss function.

Deep Learning: We exclusively employed real data and conducted training following the methodology established in the Transfer Learning paradigm for the target domain. It is important to note that the learning rate for the deep learning model was also determined in advance through a sparse grid search, resulting in a value of 0.001.

Deep Learning - Warm Start: Exclusively utilizing real data, we followed the training approach of the target domain as outlined in the Transfer Learning methodology. In contrast to regular Deep Learning, the network was initialized only before the initial training; subsequently, parameters from the previous training were retained throughout the rolling training without further initialization.

Deep Learning - Expanding Window: The Expanding Window method refers to a strategy for incrementally increasing the size of the dataset used for training a model. As time progresses, new data is continuously incorporated into the training set, expanding the window of historical data the model learns from.

Pooling Data Method/Bayesian Method: We merged the synthetic data from the source domain with the empirical data set, aligning this approach with the core techniques of Bayesian Vector Autoregression. Subsequently, training was executed following the target domain's methodology without pretraining.

Physics-Informed Neural Network Method: Physics-Informed Neural Networks typically integrate governing equations into neural training by adding penalty terms. In our experiments, we implemented a similar approach. Specifically, we modified the training loss function by first computing the weighted mean absolute error (MAE) between predicted and actual prices, as well as the MAE between predicted and model-derived prices. We then determined the weighting coefficients through an in-sample grid search.

Transfer Learning with No-Arb Constraint: We adopted the training pattern of Transfer Learning, with the distinction that the output variable was no longer the price derived from the Black-Scholes model. Instead, it ranged between randomly generated upper and lower bounds under the no-arbitrage conditions. The upper bound was defined as Ke^{-rt}, while the lower bound was $max(Ke^{-rT} - S, 0)$, where r represented the risk-free rate, S denoted the current price of the underlying asset, K was the option's strike price, r stood for the risk-free rate, and T represented the remaining time to expiration.

Transfer Learning with Only Input Variables from the Black-Scholes Model: Within the Transfer Learning framework, we retained solely the inputs utilized in the Black-Scholes model. This means that during the training in both the source domain and the target domain, only the inputs of the Black-Scholes model were considered as inputs for the neural network.

Deep Learning with Only Input Variables from the Black-Scholes Model: In the context of Deep Learning, only the input used in the Black-Scholes model were retained.

B Description of Features for DL and TL

1. **S**: Represents the price of the S&P 500 index divided by 1000.
2. **K**: The strike price of an option divided by 1000, which is the price at which the holder of the option can buy (call) or sell (put) the underlying stock or index.
3. **T**: Time to expiration, expressed in years. This is the remaining time until the option expires.

4. **r**: The risk-free interest rate.

5. **d**: The dividend yield, which is the rate of dividends paid out by the underlying stock or index relative to its price.

6. **IV1**: Lagged 1-day of the VIX index. The VIX is referred to as the market's "fear gauge" and measures the market's expectation of volatility over the upcoming 30 days.

7. **mom1w**: Short-term momentum of the option price calculated as the logarithm of the ratio of the real price one day ago to the real price six days ago.

$$\text{mom1w} = \log\left(\frac{\text{real\_price}_{t-1}}{\text{real\_price}_{t-6}}\right)$$

8. **mom4w**: Medium-term momentum of the option price calculated as the logarithm of the ratio of the real price one day ago to the real price 21 days ago.

$$\text{mom4w} = \log\left(\frac{\text{real\_price}_{t-1}}{\text{real\_price}_{t-21}}\right)$$

9. **hv1m**: This variable represents the annualized standard deviation of the daily log returns of the S&P 500 index, calculated over a 20-day rolling window. The calculation involves taking the square root of the sum of squared log returns over the past 20 days, multiplying by the annualization factor (252 trading days in a year), and then dividing by the window size (20 days). This value is lagged by one day:

$$\text{hv1m} = \sqrt{\left(\frac{\text{Sum of squared log returns over 20 days} \times 252}{20}\right)}$$

10. **hv9m**: Similarly, this variable calculates the annualized standard deviation of daily log returns over a 180-day rolling window. The procedure is the same as for hv1m, but using 180 days for the window size. This value is also lagged by one day:

$$\text{hv9m} = \sqrt{\left(\frac{\text{Sum of squared log returns over 180 days} \times 252}{180}\right)}$$

11. **volume5d**: 5-day moving average of daily abnormal volume, lagged by one day, where abnormal volume is defined as contract i's trading volume on date t divided by the average

daily volume per contract across all contracts over the preceding three months.

12. **volume1m**: Similar to volume5d, this is the 20-day moving average of daily abnormal volume, lagged by one day.

13. **Put-Call Ratio (PCratio)**: The ratio of total open interest of puts to calls across all SPX options at the end of the day, lagged by one day.

$$\text{PCratio} = \frac{\text{Open Interest of Puts}}{\text{Open Interest of Calls} + 1}$$

14. **Earnings-Price Ratio**: This ratio is calculated as the earnings (E) divided by the price (P) of S&P 500.

15. **spmom1w**: The logarithm of the short-term momentum of the S&P 500 index, measured as the ratio of the index value one day ago to six days ago.

$$\text{spmom1w} = \log\left(\frac{\text{price}_{t-1}}{\text{price}_{t-6}}\right)$$

16. **spmom4w**: The logarithm of the medium-term momentum of the S&P 500 index, measured as the ratio of the index value one day ago to 21 days ago.

$$\text{spmom4w} = \log\left(\frac{\text{price}_{t-1}}{\text{price}_{t-21}}\right)$$

C Robustness

C.1 Network Hyperparameters

In the source domain, we define the hyperparameters for multi-objective training. The weights assigned to pricing, option Delta, and Vega are set at 10:2:2. The guiding principle behind these settings is to prioritize minimizing pricing errors.

An immediate inquiry that arises is the sensitivity of the results, as derived from the transfer learning algorithm presented in this study, to the weight settings of the multi-objective optimization. To address this, we conducted a robustness test. Initially, we diminished the

Table A.1: Distributions for Source Domain Feature Generation

This table specifies the distribution type and ranges used for generating input features in the synthetic data.

Feature	**Distribution**	**Range**
S	Uniform	0 to 5
K	Uniform	0 to 5
T	Uniform	0 to 4
r	Uniform	0 to 0.1
d	Uniform	0 to 0.1
IV1	Uniform	0 to 1
mom1w, mom4w	Uniform	-10 to 10
hv1m, hv9m	Uniform	0 to 0.8
volume1m, volume5d	Uniform	-4 to 4
PCratio (Put-Call Ratio)	Uniform	0 to 2.5
epratio (Earnings-Price Ratio)	Uniform	0 to 0.1
spmom1w, spmom4w	Uniform	-0.3 to 0.2

weight for pricing, adjusting the weights to a 10:4:2 ratio, and then assessed any notable shifts in the implied volatility-median absolute error curve. Moreover, we augmented the weight for Vega hedging, calibrating the weights to a 10:2:0 ratio, and subsequently examined the implied volatility-median absolute error curve.

From the observations in Figure A.1, variations in the weight assignments of the source domain exert minimal impact on the error curve, confirming the robustness of our findings. The error curves for different weight configurations display remarkable overlap at several time nodes. The mean disparity between the error curve, when the source domain multi-objective learning weights are set at 10:4:2, and the primary outcome's error curve stands at a mere 2.51×10^{-4}. With the weights adjusted to 10:2:0, the average deviation between the error curve and the principal result is 7.37×10^{-3}. Neither of these values carry significant weight either economically or statistically. Thus, the outcomes of the transfer learning derivative pricing model display resilience against alterations in the source domain multi-objective learning weight settings. If we compare the slight differences, the 10:2:0 configuration performs relatively the worst among the three, which indicates that although hedging and pricing might ostensibly appear as divergent tasks, each inherently informs the other. The gradient of the pricing objective function essentially forms the hedging objective function. Consequently,

Table A.2: **Explanatory Variable Description For Attribution Analysis**

The table reports the observation level and definitions of the explanatory variables used in regression (16) and Table 2.

Variable	Observation	Description
0-7	Contract-daily	Dummy for time to maturity within 7 days
8-14	Contract-daily	Dummy for time to maturity between 8 and 14 days
90+	Contract-daily	Dummy for time to maturity or 90 days or more
DOTM	Contract-daily	Dummy for deep out of the money: $m \leqslant -4$
OTM	Contract-daily	Dummy for out of the money: $-4 < m \leqslant -2$
ITM	Contract-daily	Dummy for in the money: $2 \leqslant m < 4$
DITM	Contract-daily	Dummy for deep in the money: $m \geqslant 4$
BAspread	Contract-daily	Bid-ask spread
VIX60	Daily	60-day average of VIX
ΔVIX	Daily	Difference in VIX between date t and training-sample average
abnvolume	Daily	Cross-sectional average of volume5d over all contracts on date t
distance	Contract-daily	Mahalanobis distance between each contract observation and the distribution of the training set

the weight parameter for multi-objective learning merely necessitates that the neural network duly emphasizes both aspects. Intriguingly, augmenting the emphasis on pricing errors in the source domain doesn't necessarily correspond to a reduction in out-of-sample pricing errors in the target domain. At first glance, this might seem paradoxical. How can the pricing error increase by 7.37×10^{-3} upon lessening the weight assigned to the vega hedging error? Conventionally, one might posit that if a neural network's primary aim is pricing, its loss function should exclusively account for pricing errors. Yet, our empirical observations underscore that the model's efficacy in the target domain is enhanced by engaging in multi-task learning within the source domain. Overlooking hedging factors in the source domain might inadvertently diminish the precision in pricing. This intertwined relationship can be attributed to the inherent nexus between pricing and hedging. The insights a neural network garners from the structured hedging model invariably aid its pricing endeavors.

C.2 Alternative Neural Networks

The structure presented in this paper offers significant flexibility in the choice of neural network architecture, and its academic relevance remains undiminished irrespective of advancements

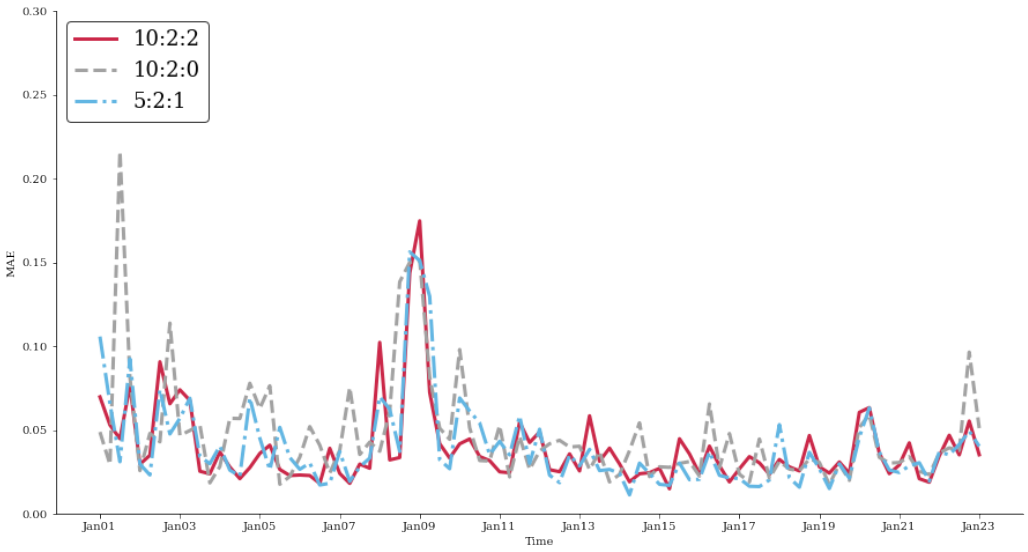

Figure A.1: **Robustness Check: Weight of Source Domain Multi-target Training.** We change the Weight of Source Domain Multi-target Training. The main results in the main text report the pricing, delta hedging, and vega hedging weights set to 10:2:2. We set the weights to 5:2:1 and 10:2:0 to observe whether there is a large change in the pricing error. The frequency of error evaluation is the same as the frequency of model retraining, that is, model retraining, model validating, and model evaluation every three months. The loss function of the model is chosen as IV-MAE. Specifically, we first convert the model-predicted price and the real price into implied volatility and then calculate the average absolute error between the two implied volatility. This scheme is designed to ensure that the model pays sufficient attention to out-of-the-money options.

in deep learning technology.

In exploring the types of neural networks, we've delved into two main aspects: adjusting the neural network's activation function and altering its structure. Initially, this study substitutes the LeakyReLU activation function with the ELU function in the primary results. Both ELU and LeakyReLU are variations of the ReLU function. While LeakyReLU enhances the gradient when the neuron input's linear sum is negative, ELU primarily aims to produce a more robust learning process by combining the properties of linear units for positive inputs and exponential units for negative inputs. The ReLU function is non-differentiable at x=0. In contrast, ELU closely resembles ReLU but helps to mitigate the vanishing gradient problem by ensuring that the function is differentiable at all points and smoothly transitions between

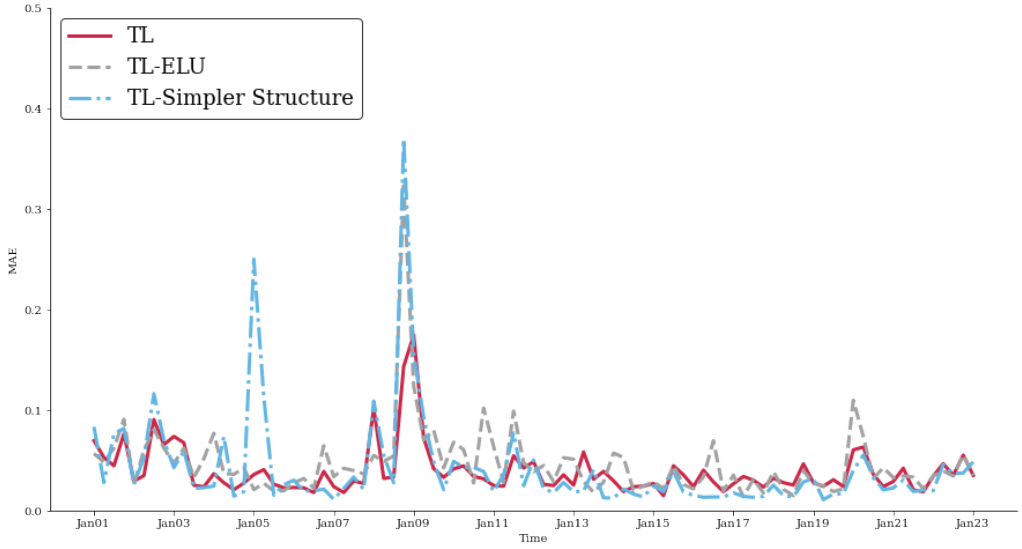

Figure A.2: **Robustness Check: NN structure and activation function.** We change the activation function. The frequency of error evaluation is the same as the frequency of model retraining, that is, model retraining, model validating and model evaluation every three months. The loss function of the model is chosen as IV-MAE. Specifically, we first convert the model-predicted price and the real price into implied volatility and then calculate the average absolute error between the two implied volatility. This scheme is designed to ensure that the model pays sufficient attention to out-of-the-money options.

the linear and exponential segments. Our objective is to discern how this smoother activation function impacts outcomes. In the subsequent figure, we examine the model's error outcomes and the core regression results from the attribution analysis after replacing all instances of LeakyReLU with the ELU function. In the error analysis presented in Figure A.2, ELU-Transfer learning exhibits similarities to the primary result's error curve. The mean deviation between ELU and LeakyReLU's error curves is 7.34×10^{-3}, economically negligible.

These findings suggest that the error curve remains robust amidst activation function alterations. Modifying the neural network's computational unit type doesn't influence the algorithm's economic rationale or empirical outcomes, aligning with our foundational hypothesis. This paper's core algorithmic design accentuates neural network performance by leveraging the inherent information of the economic model, which serves as an anchoring mechanism. Thus, the methodology here should be broadly applicable across diverse neural

networks. However, this paper's discourse on neural network activation function selection is not exhaustive. Hypothetically, each neural network layer could feature distinct activation functions. Given the myriad activation function choices, a comprehensive discussion would entail evaluating K^N scenarios, where K represents the activation function candidate set and N signifies the neural network layers count. If we consider varying the layer count, the scenarios become theoretically infinite. Our dialogue here is not to enumerate all activation function permutations but to underline this paper's methodological universality.

The activation function within the neural network can be perceived as its micro-level attributes, while the overarching design of the network represents its macro-level characteristics. The realm of computer science presents a plethora of structural designs for this aspect. In our primary study, we employed a residual learning structure. Distinct from traditional deep learning algorithms, this structure incorporates direct channels between layers to mitigate the vanishing gradient dilemma. The network we discussed ensures a direct connection between any two consecutive layers. We also explored scenarios where some direct connections were omitted. In the absence of some direct channels, the network reverts to the simpler multi-layer perceptron design.

The outcomes of this configuration are detailed in Figure A.2. Empirical findings suggest model robustness irrespective of the neural network's architecture. Notably, a non-residual learning structure appears to underperform in time series analyses when juxtaposed against its residual learning counterpart, but by only 2.11×10^{-3} on average.

This uptick can be attributed both to the trimmed network layers and the intrinsic issues non-residual learning might introduce, such as vanishing or exploding gradients. However, even in scenarios where all layers don't incorporate residual learning, our algorithm consistently outperforms conventional techniques. This superior efficacy stems from a synergistic blend of knowledge- and data-driven methodologies, resulting in impressive pricing capabilities. Concurrently, it's pivotal to recognize the invaluable contributions from the computer science domain. Superior network architectures also tend to exhibit enhanced performance under the umbrella of transfer learning. This insight further accentuates that our transfer learning-based derivatives pricing algorithm benefits from the integration of advanced artificial neural network techniques.

Parallel strategies can be seamlessly applied across a variety of network architectures, including LSTM, traditional multi-layer perceptrons (MLP), GRU, convolutional neural networks (CNN), self-attention mechanisms, Transformers, and other architectures. These strategies enable effective implementation across diverse neural network structures to address the needs of option pricing and hedging. Even as the computer science community introduces more sophisticated neural network structures, future scholars can leverage the methodology articulated in this paper to devise transfer learning models for pricing and hedging based on these novel architectures, potentially achieving superior results.

D Feature Importance in Financial Crisis

During the 2008-2009 Financial Crisis, the feature importance landscape differs markedly from the full-sample analysis. Specifically, the S&P 500 index price (S), the option's strike price (K), and the risk-free interest rate (r) all show a notable decline in their relative influence. The extreme volatility and sudden market movements characteristic of this period tend to overshadow these standard option-pricing factors, thereby reducing their impact on the model's performance.

By contrast, both the short-term ($hv1$) and long-term ($hv9$) historical volatility measures exhibit relatively higher importance compared to other features. In a market dominated by abrupt fluctuations and heightened uncertainty, accurately modeling historical volatility becomes critical, explaining why these volatility-related features take center stage. Meanwhile, the put-call ratio ($PCratio$), typically an informative gauge of market sentiment, experiences a marked decrease in explanatory power. Under crisis conditions, dramatic and fast-paced shifts in market sentiment often render the incremental information from the put-call ratio less pivotal than volatility measures.

Overall, these shifts emphasize how rapidly changing volatility dynamics can overshadow conventional pricing inputs during turbulent times. While variables like S, K, and r are fundamental in stable environments, the crisis period demonstrates that capturing volatility accurately is paramount when markets become erratic.

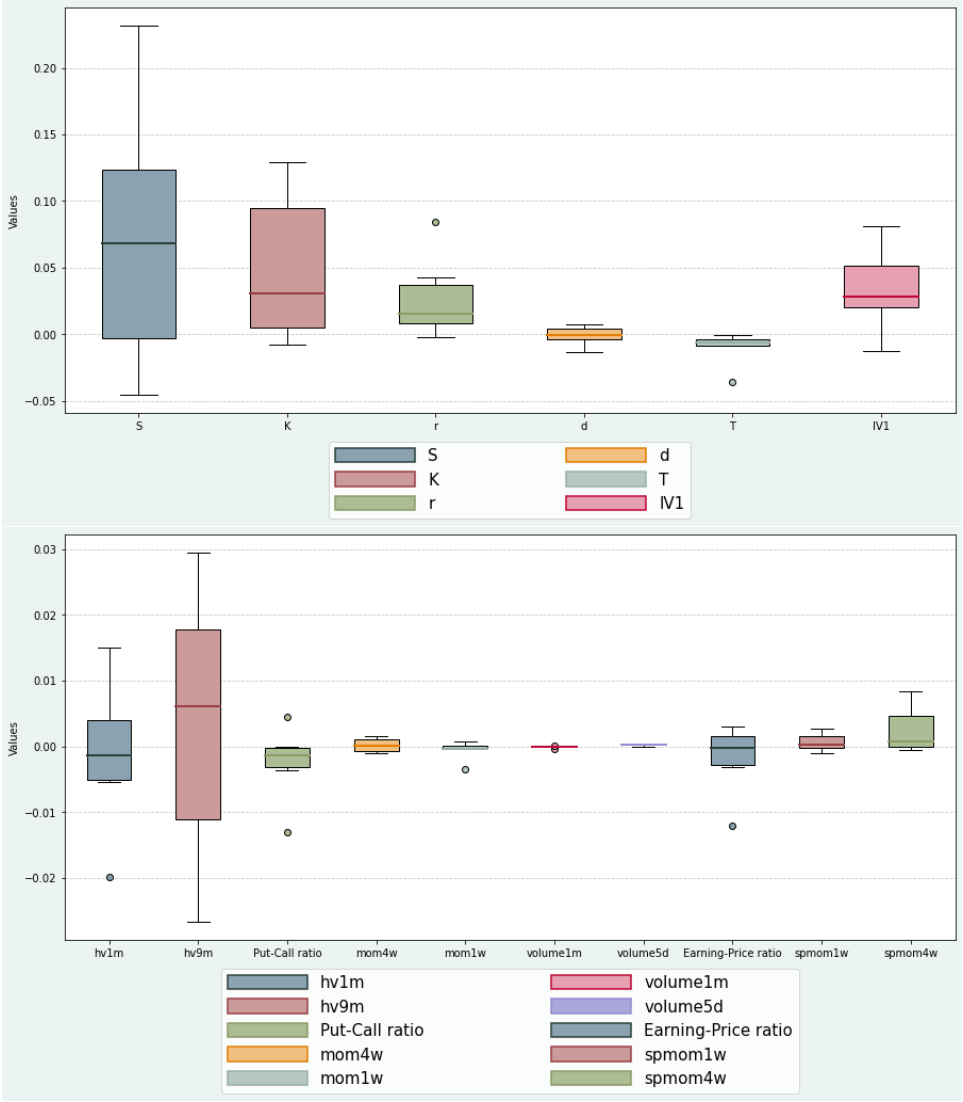

Figure A.3: **Feature Importance in Financial Crisis.** The figure below shows a boxplot of feature importance for a transfer learning model in the period of Financial Crisis. We use the data from the first quarter of 2008 to the third quarter of 2009 for calculation. We have calculated the feature importance of the same feature in each training set. This figure aims to visualize the comprehensive features of feature importance on different training sets. The outlier cut-off points of the boxplots were set to Q1-1.5IQR and Q3+1.5IQR. Data falling outside the cutoff point for outliers are marked separately.

References

Almeida, C., J. Fan, G. Freire, and F. Tang. 2023. Can a machine correct option pricing models? *Journal of Business & Economic Statistics* 41:995–1009.

Andersen, T. G., N. Fusari, and V. Todorov. 2017. Short-term market risks implied by weekly options. *Journal of Finance* 72:1335–86.

Arjovsky, M., L. Bottou, I. Gulrajani, and D. Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893* .

Bali, T. G., H. Beckmeyer, M. Moerke, and F. Weigert. 2023. Option return predictability with machine learning and big data. *Review of Financial Studies* 36:3548–602.

Bianchi, D., M. Büchner, and A. Tamoni. 2021. Bond risk premiums with machine learning. *Review of Financial Studies* 34:1046–89.

Black, F., and M. Scholes. 1973. The pricing of options and corporate liabilities. *Journal of Political Economy* 81:637–54.

Bollen, N. P. B., and R. E. Whaley. 2004. Does net buying pressure affect the shape of implied volatility functions? *Journal of Finance* 59:711–53.

Bryzgalova, S., V. DeMiguel, S. Li, and M. Pelger. 2024. Asset-pricing factors with economic targets. Working paper, London Business School.

Bryzgalova, S., M. Pelger, and J. Zhu. 2023. Forest through the trees: Building cross-sections of stock returns. Working paper, London Business School.

Campello, M., L. W. Cong, and L. Zhou. 2024. A data-driven-robust-control approach to corporate finance and ai-guided managerial actions. Working paper.

Caruana, R. 1997. Multitask learning. *Machine Learning* 28:41–75.

Chen, H., A. Didisheim, and S. Scheidegger. 2025. Deep surrogates for finance: With an application to option pricing. *Journal of Financial Economics*, forthcoming.

Chen, H., S. Joslin, and S. X. Ni. 2019. Demand for crash insurance, intermediary constraints, and risk premia in financial markets. *The Review of Financial Studies* 32:228–65.

Chen, L., M. Pelger, and J. Zhu. 2024. Deep learning in asset pricing. *Management Science* 70:714–50.

Chen, X., and S. C. Ludvigson. 2009. Land of addicts? an empirical investigation of habit-based asset pricing models. *Journal of Applied Econometrics* 24:1057–93.

Chen, X., and H. White. 1999. Improved rates and asymptotic normality for nonparametric neural network estimators. *IEEE Transactions on Information Theory* 45:682–91.

Chinco, A., A. D. Clark-Joseph, and M. Ye. 2019. Sparse signals in the cross-section of returns. *Journal of Finance* 74:449–92.

Cong, L. W., K. Tang, J. Wang, and Y. Zhang. 2022. Alphaportfolio: Direct construction through deep reinforcement learning and interpretable ai. Working paper, Cornell University.

DeJong, D., B. F. Ingram, and C. Whiteman. 1993. Analyzing vars with monetary business cycle model priors. In *Proceedings of the American Statistical Association, Bayesian Statistics Section*, vol. 160, 69.

Del Negro, M., and F. Schorfheide. 2004. Priors from general equilibrium models for vars. *International Economic Review* 45:643–73.

Doan, T., R. Litterman, and C. Sims. 1984. Forecasting and conditional projection using realistic prior distributions. *Econometric reviews* 3:1–100.

Entezari, R., H. Sedghi, O. Saukh, B. Neyshabur, and L. Dinh. 2021. The role of permutation invariance in linear mode connectivity of neural networks. In *Advances in Neural Information Processing Systems*, vol. 34, 23060–72.

Feng, G., Z. He, N. G. Polson, and J. Xu. 2023. Deep learning in characteristics-sorted factor models. *Journal of Financial Economics* 148:601–22.

Freyberger, J., A. Neuhierl, and M. Weber. 2020. Dissecting characteristics nonparametrically. *Review of Financial Studies* 33:2326–77.

Garcia, R., and R. Gençay. 2000. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics* 94:93–115.

Gârleanu, N., L. H. Pedersen, and A. M. Poteshman. 2009. Demand-based option pricing. *The Review of Financial Studies* 22:4259–99.

Geirhos, R., J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence* 2:665–73.

Gu, S., B. Kelly, and D. Xiu. 2020. Empirical asset pricing via machine learning. *Review of Financial Studies* 33:2223–73.

Hanin, B. 2019. Universal function approximation by deep neural nets with bounded width and ReLU activations. *Mathematics* 7:992–.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The elements of statistical learning: Data mining, inference, and prediction.* Springer Series in Statistics, 2nd ed. New York: Springer. doi:10.1007/978-0-387-84858-7.

He, K., and J. Sun. 2015. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5353–60.

He, K., X. Zhang, S. Ren, and J. Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–8.

———. 2016b. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, 630–45. Springer.

Heston, S. L. 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies* 6:327–43.

Hornik, K., M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2:359–66.

Hutchinson, J. M., A. W. Lo, and T. Poggio. 1994. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance* 49:851–89.

Ingram, B. F., and C. H. Whiteman. 1994. Supplanting the 'minnesota' prior: Forecasting macroeconomic time series using real business cycle model priors. *Journal of Monetary Economics* 34:497–510.

Kelly, B., and S. Pruitt. 2013. Market expectations in the cross-section of present values. *Journal of Finance* 68:1721–56.

Kirkpatrick, J., R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114:3521–6.

Kozak, S., S. Nagel, and S. Santosh. 2023. Shrinking the cross-section. *Journal of Financial Economics* 147:335–62.

Li, X., Y. Grandvalet, and F. Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. In J. Dy and A. Krause, eds., *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, 2825–34. PMLR.

Litterman, R. B. 1986. Forecasting with bayesian vector autoregressions—five years of experience. *Journal of Business & Economic Statistics* 4:25–38.

Lundberg, S. M., and S.-I. Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 30, 4765–74.

Raissi, M., P. Perdikaris, and G. E. Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378:686–707.

Rapach, D., and G. Zhou. 2013. Forecasting stock returns. In *Handbook of economic forecasting*, vol. 2, 328–83. Elsevier.

Shapley, L. S. 1953. A value for n-person games. In H. W. Kuhn and A. W. Tucker, eds., *Contributions to the Theory of Games*, vol. 2, 307–17. Princeton, NJ: Princeton University Press.

Srivastava, R. K., K. Greff, and J. Schmidhuber. 2015. Training very deep networks. *Advances in Neural Information Processing Systems* 28.

Weiss, K., T. M. Khoshgoftaar, and D. Wang. 2016. A survey of transfer learning. *Journal of Big Data* 3:9–.

Zhang, C., S. Bengio, M. Hardt, B. Recht, and O. Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*.

Zhuang, F., Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE* 109:43–76.